GÖTEBORGS UNIVERSITET

Programmering som ett verktyg för lärande -

Lärares uppfattningar om programmeringens bidrag till andra ämnen

Programming as a learning tool -

Teachers' view of the contribution of programming to other subjects

Anders Gerestrand

Magisteruppsats i Tillämpad IT med inriktning mot lärande och kommunikation

Rapport nr. 2017:103

Göteborgs universitet
Institutionen för tillämpad informationsteknologi
Göteborg, Sverige, juni 2017

Abstrakt

Skolverket har fått i uppdrag av regeringen att införa programmering i skolan, framförallt i matematikundervisningen. Papert (1984) har beskrivit programmering som ett tankeverktyg. Han hade en vidare syn än att bara tillämpa det i matematikundervisningen, även fast mycket av hans forskning var på matematik. Frågan blir då hur kan programmering användas som ett tankevertyg utanför matematiken? För att undersöka detta har lärare fått ge sin uppfattning i att använda programmering som en del i undervisningen, de har då frågats om möjligheter och utmaningar. Men även hur undervisningen kan utformas, fokus har då legat på att se på överföringseffekter till andra ämnen. Datainsamlingen har skett genom en fokusgrupp, där lärarna har fått reflektera över programmeringsövningar samt en enkät där lärarna har fått beskriva sin syn. Dessa båda metoder följdes upp av en intervju för att utvärdera data. Uppsatsen visar att när lärarna har lärt sig att programmera och implementerat det i sin undervisning är de mer positiva till programmering. De ser även fler möjligheter att använda programmering på olika sätt. Dels kan de variera själva programmeringsundervisningen och dels kan de använda det som ett lärverktyg i andra ämnen. Dock har de svårt att beskriva konkret dessa sätt självständigt. Men fick de resonera kring konkreta övningar såg de flera möjligheter. Till hjälp för denna process har det presenterats en modell som klassificerar programmeringsövningars möjligheter till att bidra till andra ämnen. Modellen har 5 nivåer: ren programmering, programmering + miljö, delar i datalogisk tänkande, programmering som verktyg, programmering för att upptäcka samband. Uppsatsen ska ses som en förstudie och skriver fram ett möjligt angreppssätt på frågan.

Nykelord
Programmering, datalogiskt tänkande, transfer, procedural retorik skola, IKT, lärande, lärare, digitala verktyg, fokusgrupp, intervju, enkät

Handledare Berner Lindström
Examinator Alexandra Weilenmann

Abstract

The National Agency for Education has been commissioned by the government to introduce programming at school, especially in mathematics education. Papert (1984) has described programming as a thinking tool. He had a wider vision than just applying it to the mathematics education, although much of his research was in mathematics. The question then becomes, how can programming be used as a tool for thinking beyond mathematics? To investigate this, teacher's thoughts of using programming as part of the teaching have been collected. They have been asked about opportunities and challenges in this subject. They have also been asked how the teaching in programming can be designed, focus has been on looking at transfer effects to other subjects. The data collection has taken place through a focus group, where the teachers have reflected on programming exercises and a questionnaire where the teachers have describe their views. These two methods were followed by an interview to evaluate the data. The result shows that when teachers have learned to program and implement it in their teaching, they are more positive about programming. They also see more opportunities to use programming in different ways. For example to vary the programming education, and they can also use it as a teaching tool in other subjects. However, they have difficulties to describe exactly how to do it. But when they have been shown different exercises they saw more possibilities. To facilitate this process, a model has been presented that classifies the ability of programming to contribute to other subjects. The model has 5 levels: pure programming, programming + environment, parts of computational thinking, programming as tool, programing to discover connection. The essay is to be seen as a preliminary study and a possible approach to this area.

Keywords
Programming, computational thinking, transfer, procedural rhetoric, school, ICT, learning, teacher, digital tools, focus group, interview, questionnaire

Supervisor Berner Lindström
Examiner Alexandra Weilenmann

Denna uppsats i Smålands skrevs
medan håret på huvudet revs.
Det klurades på programmeringens bidrag,
om till andra ämnen är ett genidrag.
Min handledare Berner visade vägen
när vi mot målet vi strävade trägen.
Även alla ni andra som med kommentarer och tankar,
era värdefulla åsikter jag högt rankar.
Nu jag målet har nått
och många nya kunskaper fått.

Innehållsförteckning

1. Inledning

Regeringen genom Helene Hellmark Knutsson gav Skolverket 2015 i uppdrag att utreda hur den nationella it-strategin ska se ut (Hellmark Knutsson, 2015). I uppdraget hänvisar de till det som digitaliseringskommissionen föreslår:

> Som kommissionen framför kan programmering som ett tydligt inslag i grundskolan ge eleverna grundläggande kunskaper för att kunna hantera sin digitala vardag och sina digitala verktyg, träna logiskt tänkande samt tidigt väcka både pojkars och inte minst flickors intresse och lust för tekniska frågor för att med tiden få fler att söka sig till tekniska utbildningar. (Hellmark Knutsson, 2015, sid. 7)

Skolverket har under 2016 utrett frågan men den har inte fastställts än. Däremot finns det flera dokument som beskriver deras arbete så här långt. I Skolverket (2016b) kan man läsa:

> I vissa sammanhang kan programmering vara synonymt med ”skrivande av kod” medan det i andra sammanhang avses ett vidare perspektiv på programmering där även problemformulering, val av lösning, att pröva och ompröva samt att dokumentera räknas in i programmeringsaktiviteten. För detta krävs förmåga till kreativ problemlösning, logiskt tänkande, ett strukturerat arbetssätt och förmåga att generalisera. Det senare perspektivet på programmering har varit utgångspunkt för förslagen till förändringar i styrdokumenten. Det har också varit viktigt att skriva fram hur programmering kan användas för till exempel styrning, reglering och problemlösning samt de demokratiska dimensionerna av programmering. (Skolverket, 2016b, sid. 7)

Tanken från skolverket i rapporten, är att programmering ska in som en del i matematik- och teknikämnet, men även en del i andra ämnen. ”Även ämnena samhällskunskap, svenska och svenska som andraspråk har förslag till ändringar kopplade till programmering.” (Skolverket, 2016b, sid. 11) För att detta ska kunna bli verklighet behöver omformuleringar göras i läroplanen. Nästa steg blir att lärarna ska förstå vad det innebär. Frågan kvarstår dock vad ska eleverna utbildas i?

Skolverket har valt att använda begreppet programmering istället för datalogiskt tänkande när de beskriver området. Det innebär också att de positionerar sig hur undervisningen ska se ut genom att välja vilka begrepp de använder. Ordet programmering har sitt ursprung i att skapa program. Det innebär i sin tur att det beskriver hantverket att skapa just program. Det blir då intressant när de beskriver vilka förmågor kursen Tillämpad programmering[1] utvecklar, de tar med skrivningar som ”Förmåga att reflektera över och värdera valda strategier, metoder och resultat.” (Skolverket, 2017, sid. 2)

[1] Ny kurs på gymnasiet där eleverna ska få tillämpa programmering på sitt programs kunskapsområde.
https://www.skolverket.se/polopoly_fs/1.259613!/Till%C3%A4mpad%20programmerin

De har då lämnat programmeringen och har närmat sig en mer generisk kompetens. I dessa sammanhang brukar begreppet datalogisk tänkande användas för att beskriva den mer generiska kompetens (Vaidyanathan, 2015). Det vill säga att de behöver kunna mera, än det som ryms i beskrivningen av att programmera. Det kan vara arbetsförmåga, kreativitet, analysförmåga och problemlösningsförmåga som beskriver en generisk kompetens. Frågan är då hur undervisningen ska se ut för att dessa mål ska kunna nås? Dock har Skolverket(2016b) friskrivit sig då de använder programmering i en vid bemärkelse.

1.1. Programmering som ett verktyg för lärande

Uppsatsens titel är ”Programmering som ett verktyg för lärande ”. Säljö (2005) beskriver att verktyg/redskap skapas genom en social integration. Det viktigaste verktyget är språket, genom att det tillåter oss att kommunicera. Samtidigt finns det flera som betonar att det utvidgade textbegreppet (Åkerfeldt, 2014) inte behöver vara ”skriven text” utan det kan vara andra former av produktioner. Det är här som titeln tar sitt avstamp, att programmering kan användas för att kommunicera. Nils Bor sa en gång: ”Science is not to tell us about the universe, but to tell us how to talk about the universe.” (National Research Council, 2010, sid. 14). Tanken de beskriver är att genom att använda programmering, kan eleverna få syn på och kommunicera saker som tidigare varit svårt. Om eleverna ska tex. skriva en dikt, kan de använda metoder och begrepp från programmering. Dikten har ett mål och måste struktureras på rätt sätt för att den ska väcka rätt känsla. Den kanske inte gör det på första försöket men efter lite ”bugg fixning” kan den infinna sig (National Research Council, 2010)

1.2. Problemformulering

En skillnad mellan programmering och datalogisk tänkande är vad som lärs under övningen på en programmeringslektion. Programmeringen kan ses som ett praktiskt hantverk medan datalogiskt tänkande ses som ett sätt att hantera området. Det innebär att beroende på hur vi tillämpar dessa begrepp kommer det vara en skillnad på det som lärs vid undervisningen. Förhoppningen är att lärandet inte stannar vid att bara kunna programmera, utan kan generera andra kunskaper och förmågor. I exempelvis artiklar som Flannery och Bers (2013) beskrivs det ingående hur barnen utvecklar sin kognitiva förmåga, men de resonerar inte om hur denna förmåga kan påverka andra ämnen. Det bara finns där som ett svävande moln.

Ett annat exempel är när eleverna på en lektion styr sina robotar eller programmerar i Scratch[2]. De lär sig att programmera så att rätt saker händer på skärmen/ med roboten. Men kan de ta med sig denna programmeringskunskap och tillämpa det på något annat? Det är troligt att programmeringen blir ett verktyg i just denna kontext. Det kan även vara svårt för elever att använda programmering och/eller datalogiskt tänkande självständigt.

[2] https://scratch.mit.edu/ Miljö för att lära sig programmering, främst för barn.

Programmeringskunskapen kan bli svår att använda i andra sammanhang som att lära sig något inom andra ämnen. Samtidigt kan de i andra ämnen mer eller mindre omedvetet arbeta med någon del av datalogisk tänkande. Det kan vara tex. när eleverna abstraherar det viktiga ur en text. Undervisningen borde kunna föra ihop dessa delar och synliggöra datalogiskt tänkande som en del i undervisningen. Arbetssättet borde göra datalogiskt tänkande till ett verktyg för lärande. Uppsatsen vill se på utvecklings-möjlighet, dvs. att kombinera datalogiskt tänkande med andra icke-tekniska ämnen. För att möta denna tanke kommer några programmeringsövningar föreslås, dessa analyseras och diskutera utifrån läroplanens tankar. Detta för att se om datalogiskt tänkande har potential att bli ett hjälpmedel i andra ämnen, utöver matematik- och teknikämnet.

2. Syfte

Syfte i uppsatsen är att beskriva lärares uppfattning och erfarenhet av programmeringens förutsättningar till att användas i undervisning och programmeringsövningars relevans i undervisning genom följande frågeställningar:

- Hur kan det som lärs ut via programmeringsövningar bidra till att nå kunskapskraven och centrala innehållet i ett ämne som inte explicit undervisar i programmering?
- Vilka möjligheter och utmaningar ser lärarna att införa och använda programmering som en del i sin undervisning?
- Hur kan programmeringsövningar struktureras efter deras bidrag till undervisning?

3. Teori

I detta avsnitt kommer datalogisk tänkande och programmering att diskuteras. Det börjar med en utblick mot Europa. Den leder till en historisk bakgrund som diskuteras. Olika begrepp som transfer, datalogiskt tänkande tas också upp. Avsnittet avslutas med en koppling mot undervisningens kontext. Ofta när programmering diskuteras, nämns Seymour Papert (1984) som en förgrundsfigur. Anledningen är att han var tidigt ute och diskuterade dessa frågor. Givetvis fanns det forskning[3] som genomförts innan han började som diskuterar området. I detta arbete börjar diskussionen kring programmering med hans bok Mindstorms. Det var också en av de första skrifterna som populariserade tankarna. Tidpunkten låg också bra till eftersom de första hemdatorerna började se dagens ljus. Processen har sedan fortskridit genom att studera vad andra har diskuterat kring boken. Sedan har några artiklar valts ut eftersom de är välciterade, vilket borde betyda att de är tongivande.

Det finns även några review-artiklar (Grover & Pea, 2013; Mannila et al., 2014) som har använts för att få en överblick över området samt förslag på artiklar. För att vidga området har det skett sökningar mot artiklar som har att göra med ”Datalogisk tänkande” Dock är det ett begrepp som blivit mer populärt de senare åren. Avsaknaden av ordet har gjort att det varit svårt att hitta äldre artiklar den vägen. Då har sökandet inriktats mot att se på senare artiklars referenser istället.

3.1. Introduktion till ”Datalogisk tänkande”

Mannila et al. (2014) har skrivit en rapport kring hur datalogiskt tänkande används i undervisningen i Europa. De gör där en gedigen genomgång av datalogiskt tänkande, med några exempel på hur det används praktiskt. Olika länder har en terminologi som i vissa avseende är liknade, men det finns även skillnader. En del begrepp som används är programmering, kodning, ”kodning och datorprogrammering”. Olika länder har använt ”algorithmic applications (IL), algorithmic problem solving (SK) or algorithm design and data models (HU), or algorithmic and robotics (ES). Ireland and France exclusively refer to coding.” (Balanskat & Engelhardt, 2015, sid. 27) Även Pålsson (2017) beskriver att många länder har börjat ta in programmering i läroplanen.

För att definiera datalogiskt tänkande tas begrepp som algoritm, problemlösning, logik, abstraktion, mönster och uppdelning upp (Mannila et al., 2014). Ofta (Mannila et al., 2014)lyfts det fram att eleverna ska lära sig problemlösning genom programmering, genom ett formellt och logiskt tänkande. Många av dessa begrepp behöver eleven kunna använda i andra ämnen men det nämns inte utförligt i rapporten. En variant att beskriva området är Vees (2013) perspektiv då det beskrivs som computational literacy. Hon menar att programmering styr mycket i samhället, så det är ett språk som måste behärskas. Detta visar att området är komplext och att det finns flera nyanser. Denna uppsats kommer senare att försöka definiera datalogiskt tänkande. Det som forskningen brottas med är hur det ska införas i skolan på ett adekvat sätt.

3.2. Historisk bakgrund

Ordet Computer kommer från en person som gjorde beräkningar. Ordet nämndes redan på 1600-talet för första gången enligt Oxford English Directory (2017). Sedan mitten av 1800-talet började ordet användas för att benämna en maskin som gjorde beräkningar.

[3] Se beskrivning av tidig forskning om programmering i tex. Pea och Kurland (1984) och Mayer (2004)

Som nämnts tidigare var Papert (1984) en av de första att prata om datalogisk tänkande. I sin bok ”Mindstorms” (engelsk) nämner han computational thinking på sidan 181 och i den svenska översättningen ”Tankestormar” står det datametodologiskt tänkande på sidan 207. Dock går han inte närmare in på vad begreppet innebär i det stycket. Däremot försöker han förmedla sina tankar kring begreppet genom sin bok. När Papert drog igång arbetet med LOGO, var det väldigt specifik. Att tex. under matematikundervisningen lära sig att skapa en cirkel med hjälp av programmering.

Papert var under några år och arbetade med J M Piaget i Geneve och det har han tagit ett stort intryck av. Piagets tankar utgår från ett konstruktivistiskt perspektiv eller som han (Papert, 1984) kallar det ”genetisk epistemologi” Det kan förklaras som läran om hur generering av kunskap går till. Kort går hans teori ut på, att vid lärande så eftersträvas en jämnvikt (equilibrium), barnet (som Piaget ofta utgår ifrån) tar till sig ny kunskap som passar hens kognitiva strukturer (assimilation) men ibland dyker det upp ny kunskap som inte passar in, då uppstår en ojämnvikt. Barnet måste då ändra sina kognitiva strukturer för att den nya kunskapen ska passa in (ackommodation) (ibid). Synsättet kan sammanfattas på följande sätt. Eleven ser något abstrakt som den inte förstår, sedan provar hen sig fram för att se hur det fungerar i verkligheten. Detta repeteras om och om igen, vilket leder till att förståelsen förbättras. Kunskap är dock inget som kan föras över. Utan det är en inre mental process som pågår genom att barnet interagerar med omvärlden. Barnet måste vara aktivt och ta med egna erfarenheter för att deras mentala modeller ska förändras. Det innebär i sin tur att barnen bygger på sina egna kognitiva strukturer och de bygger på vad de vet. Papert (1984) har tagit fram ett programspråk som heter LOGO som skulle bli ett verktyg för tänkande, vilket vänder sig till barn. Papert skriver att inlärning ofta består i att ”man antingen ”kan” eller ”har fel”” (sid 33), det var det han ville bort ifrån. Det är däremot en skillnad när en dator ska programmeras, eftersom det aldrig blir rätt från början. Han menar att processen att komma fram till det rätta, är fundamental. Frågan blir inte att titta på vad som blivit rätt och fel utan hur det ska lösas. Debugging är därför en viktig del i processen. För att knyta an till tesen i början på stycket: eleverna bygger på vad de kan och utvecklar programmet stegvist.

Arbetet med LOGO innebar att barnet fick programmera en sköldpadda. Den rörde sig över skärmen efter de kommandon som hade skrivits. Nedan visas ett exempel på hur sköldpaddan kan rita en triangel efter ett program (Papert, 1984).

```
TRIANGEL
FRAMÅT 100
HÖGER 120
FRAMÅT 100
HÖGER 120
FRAMÅT 100
SLUT
```

Programmet kan sedan utvecklas med loopar och funktioner. Det gör att mer avancerade mönster kan ritas på skärmen med hjälp av LOGO. Språket gör att användaren får tänka abstrakt med matematiska begrepp för att kunna få fram de figurer hen vill. Notera att programmeringsspråket är textbaserat vilket gör att barnet måste ha kommit en bit i sin läs och skrivutveckling för att det ska fungera.

En tanke som Papert (1984) väckte var tesen om QWERTY kring varför tangenterna sitter som de gör. Ett tangentbord har tangenterna i en speciell ordning. Det kommer sig av att när skrivmaskinen togs fram så trasslade sig bokstavsarmarna om man skrev för

fort. Genom att sätta bokstävernas armar som ofta hamnade efter varandra när ord ska skrivas, så långt som möjligt ifrån varandra kunde detta trassel minskas. Ganska snart löstes detta problem men då var tangentbordsuppsättningen redan etablerad vilket gjorde att ingen ville gå tillbaka till en alfabetisk ordning på tangentbordet. Papert använder detta som exempel på att vi ofta fastnar i ett historiskt tankesätt. När boken skrevs användes ofta datorer till lärande utifrån ett undervisningsteknologisk perspektiv, dvs. att lärandet betingades fram. Han vill hitta ett mer organiskt sätt att använda dem. Han hävdar att flygplan inte uppfanns genom att man studerade hur en häst och vagn fungerade, utan det var ett nytt sätt att tänka. Med boken vill han visa på detta nya sätt att tänka.

Vad går hans metod ut på? Han vill som nämnts låta barnet skapa sin egen kunskap genom ett undersökande arbetssätt. Språket LOGO har konstruerats för att ge en mening till lärandet. Genom att använda det ser barnen en mening och de kan utvecklas utifrån sina förutsättningar. Han beskriver en liknelse ”… den väsentliga inlärningsupplevelsen gäller inte att memorera fakta eller utöva färdigheter. Snarare är det att lära känna sköldpaddan, att utforska vad en sköldpadda kan och inte kan göra.” (Papert, 1984, sid. 155). I boken beskriver han ett antal exempel då barnet resonerar och prövar sig fram till olika lösningar. Ett exempel är när de ska räkna ut skillnaden mellan en yttre och inre omkrets. Han beskriver då ett avancerat exempel hur eleven kan resonera för att komma fram till en procedur för en lösning. Frågan är om ett barn kan komma fram till en sådan lösning själv? I boken betonas att programmeringen behöver ha ett gränssnitt som användaren kan förstå. När LOGO används ger resultatet ett gränssnitt som lätt förstås. Den som programmerar kan se resultatet på ett tydligt sätt. Däremot är LOGOs gränssnittet genom text svårare att förstå.

Det som är bra med ett formaliserat språk är att det blir exakt. Det kan sätta ord på en handling som annars kan vara svårt att få syn på. Papert (1984) beskriver ett exempel för att lära sig jonglera. Genom att vara konkret, formell i beskrivningen är det lättare att förmedla intentionen med övningen. När en elevs vokabulär är otillräcklig för att förklara vad hen vet, är en viktig del av inlärningsprocessen förvärv av nya ord, nya koncept och ny grammatik för att förklara det. Och när det finns en sådan nära koppling mellan de nya ord (kod), grammatiken (programspråkets syntax) och de begrepp de uttrycker (programmets semantik), gör det att elevernas möjlighet att uttrycka sig flyttas framåt.

3.3. Kritik mot det konstruktivistiska synsättet

I en artikel är Pea (1983) mycket kritisk till att använda LOGO programmering för lärande. Det han tar upp som svagheter med forskningen och LOGO är att det:

- Bygger på anekdoter, god forskning behövs.
- På tok för svårt för nybörjare, speciellt att lära sig själv.
- att LOGO inte fungerar som primus motor när det gäller utveckling av tänkande.
- såg inga effekter på elevernas planeringsförmåga efter en 1 årig-undersökning.
- oklart vad programmeringsundervisning ska uppfylla.

Han avslutar artikeln med att möjligen kan de få en kunskap om grundläggande programmering men annat lärande är svårt att påvisa. Det kan också vara möjligt att LOGO kan illustrera problemlösningsmetoder. I en annan artikel skriver Pea (1987) att skolan Bank Street Collage som Papert (1984) gjorde sina studier på var en experimentell skola, så den är inte representativ för en vanlig amerikansk skola.

Tankarna i Papert (1984) bygger på ett konstruktivistiskt perspektiv, att eleven genom att upptäcka, skapar sitt lärande. Det bygger på att eleven självständigt eller med lite hjälp ska lösa uppgiften. Socialkonstruktivistiska perspektivet har liknade utgångspunkt men då arbetar eleverna i en grupp då de löser uppgiften istället. Mayer (2004) beskriver att för att kunna nå ett konstruktivistiskt lärande måste undrvisningen vara noga anpassad. I sin artikel tar han upp tre svagheter med konstruktivistiskt lärande.

- Det fungerar inte med rent upptäcktsbaserat lärande utan det måste vara styrd.
- Lärandet blir bättre om eleverna är aktiva och läraren visar i vilken riktning eleverna ska röra sig.
- Det är svårt att få en överföringseffekt till andra områden om ett strikt upptäcktsbaserat angreppssätt används vid programmeringsträning.

Han beskriver att problemet är om eleven får för mycket frihet, så blir det svårt att veta vilken information som är viktig. Som lärare är utmaningen ”know how much and what kind of guidance to provide and to know how to specify the desired outcome of learning.” (Mayer, 2004, sid. 17) Han skriver att lite då och då vurmas det för ren upptäcktsbaserat lärande, det är som en zombie som återigen kliver upp från graven, de finns de som lyfter fram metoden men stödet är svagt.

Pea och Kurland (1984) är kritiska till Paperts arbete, de har replikerat hans studier med 50 elever, för att se om de kan få liknade resultat. Det de vänder sig emot är att det sker överföring av kunskap från programmeringsuppgifterna till andra situationer. Det kan tex. vara att eleven lär sig att ”strukturera ett problem” genom LOGO-programmering, då de i själva verket lär sig hantera kommandon i LOGO. Ett annat exempel är att de lär sig matematiska koncept genom att skapa en spiral i LOGO. En annan tanke som de tar upp är om problemlösning är generellt oberoende eller är en sammankoppling av specifika delar för en speciell situation. I det första fallet kommer alla problemlösning träning vara bra men i det andra fallet behöver tränings problemen anpassas, så att rätt delar tränas in. I artikeln hävdar de att det finns lite stöd för att eleverna får en generell oberoende problemlösningsförmåga från programmeringsundervisning. Vilket gör att de menar att eleven blir bra på det hen tränar på.

Pea och Kurland (1984) beskriver även att det finns sex nödvändiga kognitiva förmågor som eleven behöver utveckla för att kunna programmera. Dessa är:

- Matematisk skicklighet.
- Analogisk resonerande - förmåga att koppla ihop kunskap och strategier både till programmeringsarbetet och utanför själva programmeringen.
- Processkapacitet - förmåga att kunna hantera variabler och koncentration.
- Villkorsresonerade - en förståelse för hur villkor i programmering fungerar.
- Procedural förmåga - kunna tänka sig hur flödet i programmet ser ut.
- Temporalresonerade - att kunna forma delar i programmet.

Dessa punkter visar att det finns kognitiva förmågor som eleven måste ha eller i alla fall utveckla för att kunna programmera.

3.4. Mer kring datalogiskt tänkande

Dalbey och Linn (1985) beskriver i sin artikel vad som krävs för att ta fram ett program. Studien är en review-artikel där de flesta ingående artiklarna har analyserat vad en programmerare gör. De har gått igenom artiklar och kommit fram till att processen beskrivs med hjälp av fyra olika stadier:

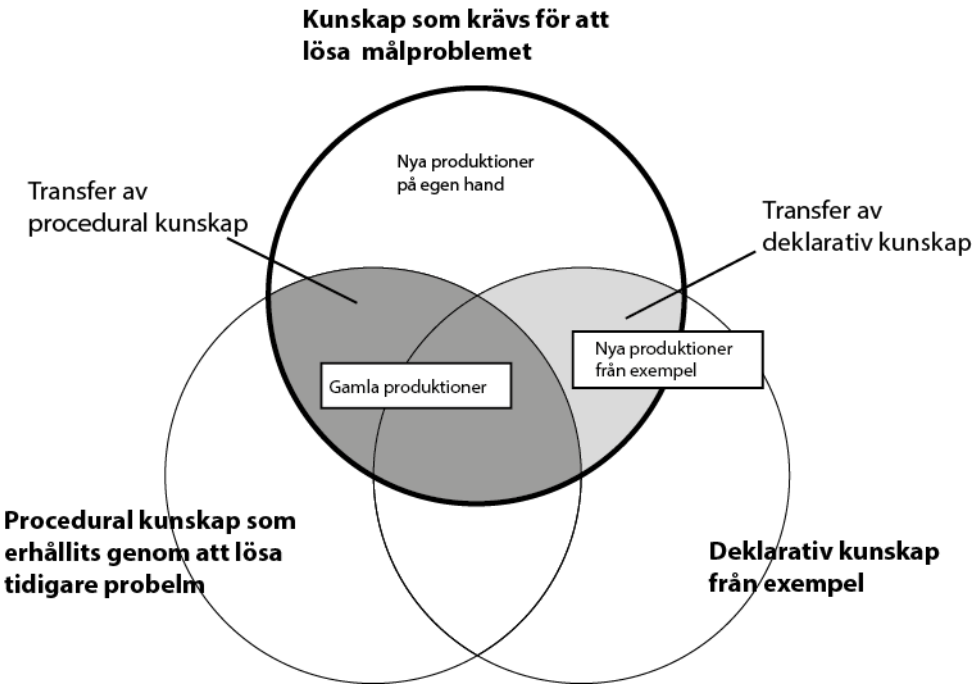
- Specifikation
    - Beskriver problemet. Ta fram rätt information, vilka behov finns, vad är programmerbart eller är ett analogt problem
- Design
    - En av de viktigaste delarna är nedbrytning, dvs. att dela upp problemet i delar som är hanterbara och meningsfulla. Duktiga programmerare är bra på det. Sedan finns en principen Top-down eller Bottom-up. När Topdown används börjar man att skapa ett komplext överprogram sedan skapas underprogrammen för att lösa de specifika delarna. Sedan fortsätter processen så tills hela programmet är klart. Bottom-up börjar i andra änden de börjar med små program som sedan sätts ihop till större och större delar, som tillslut bildar en helhet. Vilket leder till att hela programmet är klart. Oftast är top-down sätt vanligast, men en kombination av båda sätten är effektivast.

      Dalbey och Linn (1985, sid. 256) skriver ”most programs are not a single solution to a specific problem, but algorithms that work for a whole class of similar problems”. Denna förmåga att kunna skapa det kallas för ”procedurell resonerande” som går ut på att kunna fastställa ett antal instruktioner som ger önskad inverkan. Medan problemlösning ofta går ut på att hitta en specifik lösning. Frågan om hur lösningen nås är i fokus. De betonar även att det är viktigt att ha en ”mental modell” över hur datorn och program hör ihop. Denna förståelse är viktig för att kunna programmera. Detta visar att programmering ställer krav på att den kognitiva aktiviteten måste systematiseras noggrant, i förhållande till mycket annan problemlösning.
- Kodning
    - Kodning går ut på att implementera designen i ett programmeringsspråk. Även om programmerarna kan förklara sin tanke med ord, så kan de inte alltid tillämpa det i ett formellt språk. Att programmera innebär att ha detaljerad kunskap om ett komplext programspråks syntax och semantik. Utan denna kunskap är det svårt att programmera.
- Avlusa
    - Att kunna hantera när det blir fel är en viktig egenskap. Dalbey och Linn (1985) beskriver att så mycket som upp till 75 procent av tiden går åt för felsökning. Kärnan är att förstå feedbacken från programmet. Det kan vara både beteendet och information (tex. variabelvärden) från programmet.

3.5. Transfer

Ett av målen med denna uppsats är att studera om hur programmeringsundervisningen kan ge ett bidrag till andra ämnen. Transfer innebär att lärande kan förs över från en aktivitet i en situation till en aktivitet i en annan situation (Marton, 2006). Det innebär att det måste ses som en del av en kontext. När vi lär oss någonting sker det alltid i en speciell situation, för att kunna närma oss en ny situation med den kunskapen vi har behöver den föras över och kunna användas i den nya situationen. Marton (2006) säger att det är därför intressant att analysera skillnader och likheter mellan aktiviteter i situationer, vad är det som varierar och vad är konstant. I tidig forskning beskrevs transfer som en mental process (Marton, 2006) Att lärandet bygger på varandra tex. om man tränar addition sker en transfer till multiplikation. De tog ingen hänsyn till att situationen är viktig att ta hänsyn till. En definition skulle kunna se ut på följande sätt ”Transfer refers to the relations between what people learn and can do in different situations thanks to similarities between those situations.” (Marton, 2006, sid. 510) Idén är att den lärande ska förstå den generella principen från situation A och kunna tillämpa det i situation B. Oftast när transfer undersöks så försöker man hitta likheter men det är minst lika viktigt att förstå skillnaderna. Ifall vi ska kunna se skillnader måste vi ha upplevt något tidigare som vi kan se skillnader ifrån. ”If we were to cast the transfer issue in a pedagogical form, we would not only ask “What is learned?” and “What is transferred?,” but also “What should be learned?” and “What should be transferred?.” (Marton, 2006, sid. 532)

I Shih och Alessi (1994) beskriver de två olika typer av kunskap: procedurell och deklarativ. Deklarativ kunskap är fakta, koncept och principer, dessa ”kunskaper” måste göras om för att bli procedurell kunskap, dvs. hur de ska omsättas i praktiken. Ur ett transfer-perspektiv kan resultatet bli olika även om de utgår från samma deklarativa kunskap. Det innebär i sin tur att eleven har svårt att praktisera sin kunskap från en praktik till en annan. Det leder till ”there will be positive transfer between skills that involve the same productions and no transfer between skills that involve different productions even if based on identical declarative knowledge.” (Shih & Alessi, 1994, sid. 154). I sin studie håller de med om denna tes men säger samtidigt att en viss interaktion sker mellan deklarativ och procedurell kunskap. Ju mer kunskapen är organiserad hos den som lär sig (mental modell) desto mer underlättas transfer. De menar att deras studie stöder konceptundervisning som metod vid programmering. Med konceptundervisning menas att deklarativ och procedurell kunskap bör kombineras. Att sitta och skriva kod tränar inte programmering tillräckligt bra utan eleverna behöver även få till sig hur de skapar sina mentala modeller över datalogiskt tänkande.



Figur 1 Ideal modell över transfer och produktioner. (översatt) Pirolli och Recker (1994, sid. 240)

I figur 1 har Pirolli och Recker (1994) beskrivit hur ett enskilt programmeringsproblem kan lösas. Genom att ha löst egna problem tidigare och sett olika exempel kan de lösa nya problem på egen hand. I det som kallas nya produktioner har eleverna stött på nya problem genom olika exempel där de fått använda sina kognitiva förmågor. Men de har också fått kunskap genom att ha löst tidigare problem, det kallas för gamla produktioner. Det som blir kvar är det som kallas ”Nya produktioner på egen hand” dvs. det är här som de skapar en ny förståelse för hur produktionen ska lösas för att ett lärande ska uppstå. För att förstå programmering så behöver eleven kombinera sin mentala modell från tidigare programmering med uppgifter beskrivna genom text och exempel. Så deras uppgift blir att konstruera en mental modell som representerar den situation som beskrivs av texten. Denna konstruktion försvåras av två faktorer. För det första, texter om program och programmering är abstrakta, vilket försvårar bygget av en mental modell. För det andra, ofta används begrepp som inte eleven känner till vilket gör att det blir svårt att ta till sig och därmed få med in i sin mentala modell. Till exempel är det osannolikt att en elev direkt kan konstruera en mental modell av frasen "ett rekursivt anrop" om de aldrig tidigare har sett ett rekursivt anrop (ibid).

Ett begrepp som beskrivs är computational thinking patterns (CTP) (Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011) genom att se vilket innehåll som datalogiskt tänkande har och sedan hur det uttrycks. Tex. en idé om abstraktion uttrycks genom hur ett program är konstruerat. Det är då inte så viktigt hur det uttrycks (vilket programspråk som används) utan vad det är för tanke bakom. Däremot är det som uttrycks en indikator för tanken. Tanken kan visas genom programmeringen. De har gjort sin forskning på en spelbaserad miljö. De har där konstruerat ett verktyg som kan mäta CTP genom att analysera koden. Den har visat att vissa delar kan föras över från en situation till en annan. De har sett överföring mellan ett spel och en simulering på vissa aspekter.

Forskningen ovan visar att det är svårt att se en överföringseffekt från programmering och kan summeras med ett citat från Pea (1983)

> This idea--that programming will provide exercise for the highest mental faculties, and that the cognitive development thus assured for programming will generalize or transfer to other content areas in the child's life--is a great hope. Many elegant analyses offer reasons for this hope, although there is an important sense in which the arguments ring like the overzealous prescriptions for studying Latin in Victorian times. (Pea, 1983, sid. 2)

Dock beskriver Klahr och Carver (1988) att det finns möjligheter till transfer, då undervisningen är noga strukturerad. Ett annat perspektiv beskriver att mycket av transfer-forskning gjorts på ”direkt applicerbar transfer” (Bransford & Schwartz, 1999). Dvs eleverna ska exakt göra det som originalet visade fast i en annan situation. Men den typ av transfer som kallas ”Förberedelse för framtiden” (fff) är svår att få syn på och det är även svårt att designa studier som kan se den. Det innebär att fff ofta missas när transfer diskuteras. Problemet med fff är att läraren kan försvara sig med ”Jag undervisar för framtiden” dvs. allt de gör kan ha någon effekt. Därför måste undervisningens syfte utvärderas noggrant.

3.6. Datalogiskt tänkande populariseras

En artikel som ofta citeras är Wings (2006) artikel. Varje artikel med självaktning hänvisar till den då detta område diskuteras. I artikeln sägs bl.a. följande

> Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions. (Wing, 2006, sid. 33)

Artikeln tar sitt avstamp i hur datalogiskt tänkande kan användas för problemlösning, designa system och förståelse för hur människan förhåller sig till datorer. Hon beskriver sedan vad datalogiskt tänkande är och inte är. Kärnan i hennes resonemang är att det är ett verktyg för människor att tänka med. Det viktiga är inte att förstå datorer och programmering utan att genom datorer kunna skapa idéer. Förståelsen är på ett sätt något nödvändigt ont. Artikeln är upplagd som en debattartikel och försöker där väcka tankar i frågan. Den har inga referenser som är brukligt i vetenskapliga artiklar. Det gör att det är svårt att leda i bevis att hennes utsagor stämmer. Till exempel så beskriver hon att datorer hjälper folk att tänka men inte hur. Hennes utsagor är lite långtgående emellanåt. Grover och Pea (2013) beskriver att Wings artikel var starten för många initiativ inom området. De säger att kärnan är att man ska tänka som en datalog när man stöter på problem.

Jones (2011) debatterar Wings artikel genom att ställa flera frågor. Den första funderingen som hon har är att Wing hävdar att det är ett tankeverktyg för alla. Men tänkande är så komplext, så frågan är om verkligen allt mänskligt tänkande kan rymmas under rubriken. Nästa fundering är hur skiljer sig datalogisk tänkande från annan typ av tänkande. Vid forskning tex. så hanteras stora datamängder, som forskaren drar slutsatser ifrån på ett strukturerat sätt. Det behöver inte vara datalogiskt tänkande men i princip gör forskaren samma sak. Frågan blir då vad är unikt med datalogisk tänkande. En tredje fundering är att många problem är så komplexa att de inte går att abstrahera, därmed kan inte en dator lösa dem. Hon nämner att större etik- och moralproblem kan

vara svåra pga. den aspekten. Wing (2006) väcker en del tankar om datalogiskt tänkande men det finns en hel del svagheter med artikeln ur ett vetenskapligt perspektiv

3.7. Datalogiskt tänkande definieras på olika sätt

England var ett av de första länderna som införde programmering i grundskolans läroplan. En av aktörerna som förespråkade det var The Royal Society (2012), de beskriver datalogiskt tänkande på följande sätt i en rapport.

> Computational thinking is the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes.
> (The Royal Society, 2012, sid. 29)

Det är en allmän definition men det finns andra definitioner. Tex har ISTE (International Society for Technology in Education) definierat datalogiskt tänkande på följande sätt:

> Computational thinking is a problemsolving process that includes:
> - Formulating problems in a way that enables us to use a computer and other tools to help solve them.
> - Logically organizing and analyzing data.
> - Representing data through abstractions, such as models and simulations.
> - Automating solutions through algorithmic thinking (a series of ordered steps).
> - Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources.
> - Generalizing and transferring this problem-solving process to a wide variety of problems. (ISTE, 2011, sid. 1)

Vid en jämförelse av dessa definitioner är processen en viktig del att fokusera på. De har också en inriktning mot att vara praktiska i sin tillämpning. ISTE definition är dock mer konkret då den tar upp fler datalogiska begrepp. Grover och Pea (2013) beskriver en definition datalogiskt tänkande som de hävdar är mer allmänt accepterad på detta sätt:

> - Abstractions and pattern generalizations (including models and simulations).
> - Systematic processing of information.
> - Symbol systems and representations.
> - Algorithmic notions of flow of control.
> - Structured problem decomposition (modularizing).
> - Iterative, recursive, and parallel thinking.
> - Conditional logic.
> - Efficiency and performance constraints.
> - Debugging and systematic error detection. (Grover & Pea, 2013, sid. 39)

Den är dock för de som är insatta i datalogiskt tänkande och den är svår att omsätta i praktiken för tex. en lärare, då de olika begreppen inte förklaras eller definieras.

Samtidigt framgår det inte om alla punkter måste vara med eller om det räcker med bara en eller några få punkter för att det ska vara datalogiskt tänkande. Lee et al. (2011) har försökt att konkretisera processen genom att arbeta med delarna abstraktion, automation och analys. Genom dessa begrepp försöker de förstå hur unga kan använda datalogiskt tänkande för att lösa nya problem. Abstraktion innebär att man ska fånga kärnan, karaktären i problemet så att det blir en ny instans. Automation är när datorn får göra flera repetitiva uppgifter så att människan slipper göra dessa. Lite slarvigt menar de att program är automatiska abstraktioner. Analys går därför ut på att se om programmet gör det som det ska. Lee et al. (2011) beskriver en tillämpning, där ungdomar får använda simuleringar för att förstå epidemiologi. De tittade på hur spridningen påverkades av bla. hur skolan var byggd. De implementerade olika faktorer i en simuleringsmodell (abstraktion) sedan lät de datorn räkna ut spridningen över tid (automation). Därefter kunde de reglera ingångsvärden för att se vad det var som påverkade spridningen. De kunde ställa sig frågor som ”Är modellen korrekt?” (analys). De har använt ett verktyg som heter StarLogo[4] där eleverna kan utveckla detta. De rekommenderar att man ska använda en modell som de kallar Use-Modify-Create. Där den första nivån låter eleven få arbeta med färdiga program. På andra nivån får eleven förändra färdiga program och på den tredje och sista nivån får eleven skapa nya program, genom en iterativ process. Den iterativa processen består av delarna testa-analysera-omdefiniera.

Detta avsnitt kan det sammanfattas som att datalogiskt tänkande är en tankeprocess som finns beskrivna enligt tabell 1. Tabellen är hämtad från Angeli et al. (2016), de tycker att dessa egenskaper ska tränas genom att deras delar ska lyftas fram i undervisningen, de ska bockas av. Samtidigt betonar de ett holistisk sätt dvs. att dessa delar ska sättas ihop till en helhet.

Abstraktion	Kunskapen att bestämma vilken information som ska behållas/ignoreras
Generalisering	Kunskapen att formulera en lösning på ett generellt sätt så att den kan tillämpas på olika problem
Nedbrytning	Kunskapen att bryta ner ett komplext problem i mindre delar så att det blir lättare att förstå och lösa
Algoritmer	Kunskapen att utforma en stegvis uppsättning steg i rätt ordning för att kunna lösa ett problem
Felsökning	Kunskapen att identifiera, ta bort och åtgärda fel

Tabell 1 Sammanfattning av datalogiskt tänkande från Angeli et al. (2016)

Barr och Stephenson (2011) ger en ram för hur ett ämne kan använda sig av datalogiskt tänkande. Tänk dig att eleverna ska skriva en berättelse. De börjar med att samla in material till den (Data insamling), sedan bestämmer de sig för vilket material som ska finnas med (Dataanalys), eleverna namnger olika karaktärer i berättelsen (Datarepresentation). Sedan börjar eleverna skriva ett utkast (dela upp problem). Berättelsen behöver även rättstavas i programmet (Automation). Inom språkämnet kan de även behöva skriva instruktioner (algoritmer & procedurer). Eleverna kan även

[4] http://www.slnova.org/ Simuleringsverktyg

behöva skriva ett referat (Abstrahera). Denna modell visar att det skapas en god koppling mellan ett annat ämne och datalogiskt tänkande. Den menar också lärare behöver ges möjlighet att utveckla sin undervisning mot datalogiskt tänkande och då behöver lärare relevanta resurser, och förslag på aktiviteter och kursmaterial.

3.8. Kursplansanalys

Utgår man från teoriavsnittet finns det tankar som lyder att programmering är ett tankeverktyg som låter eleven få syn på kunskapen (Papert, 1984; Wing, 2006). Givetvis är det så att vissa delar är enklare att omsätta i praktiken, tex. i matematik är det lite lättare även om det har visat sig att tillämpa kunskapen till andra situationer är svårare. Men utgår vi från samhällskunskap, hur kan då programmeringen bli ett tankeverktyg för att få syn på kunskapen?

I kursplanen finns det ”centrala innehållet”, för samhällskunskap åk 7-9 finns följande rad inskriven. Det är en del av innehållet som syftar till att exemplifiera hur ”centrala innehållet” kan se ut. Anledningen att just denna läroplan valdes var att Samhällskunskap inte naturligt kopplas till programmering/datalogiskt tänkande. Men en annan kursplan kunde lika gärna ha valts.

> Arbetsmarknadens och arbetslivets förändringar och villkor, till exempel arbetsmiljö och arbetsrätt. Utbildningsvägar, yrkesval och entreprenörskap i ett globalt samhälle. Några orsaker till individens val av yrke och till löneskillnader. (Skolverket, 2011)

Som alltid är det lärarens uppgift att hitta övningar och aktiviteter som kan täcka det centrala innehållet. Det är dock intressant att leka med tanken hur programmering kan bidra till en förståelse till området. Spontant känns det inte som att det finns några naturliga öppningar för det.

3.9. Programmeringsundervisningens design

I en enkätstudie av Sentance och Csizmadia (2015) undersökte de lärarnas tankar om hur programmering skulle undervisas. Studien visade att lärarna ville att aktiviteterna i klassrummet inte skulle innefatta en dator. Det skulle vara aktiviteter där eleverna arbetar praktiskt. Men de ska ändå vara inriktade mot datalogiskt tänkande och de betonade vikten av mycket träning. I en annan artikel menar Burke (2012) att det är viktigt med ett formaliserat språk för att vara tydlig med det som är intentionen med programmet. Det blir svårare vid ”unplugged”-aktiviteter. Han presenterar även en modell (tabell 2) i tre steg (1) Produkt Vad lärde de sig? (2) Process Hur lärde de sig? (3) Perception Attityder mellan ämnet och programmering. Sedan tillämpas dessa tre steg på programmering och ämnet parallellt.

Vad är i fokus?	Scratch	Skriva
(1) Produkt	Programmerings koncept (loopar, händelser, hantering, parallellhantering)	En berättelses delar (anslag point of no return, karaktärer, konflikt)
(2) Process	Design, problemhantering, debuggning	Draft, revidering, redigering
(3) Perception	Förstå kodningen som en serie sekvenser	Förstå skrivandets karaktär och område.

Tabell 2 Jämförelse mellan programmering och skrivning. Översatt (Burke, 2012, sid. 127)

En likande modell presenterades av Benton, Hoyles, Kalas, och Noss (2017) som beskriver ett ramverk med 5 delar hur datalogiskt tänkande kan läras ut. Där den första punkten är (1) Utforska - eleverna ges möjligheten att undersöka idéer genom att själva försöka och hitta fel. (2) Förutse - Eleverna bör uppmuntras att förutsäga resultat innan de kör sitt skript och sedan reflektera över det faktiska resultatet. (3) Förklara - Eleven bör ges möjlighet att diskutera med sina kamrater i en diskussion som leds av läraren. (4) Utbyte - De ska kunna dela och bygga på varandras idéer. (5) Bro - Bra det ska vara en tydlighet mot som i det här fallet matematikens läroplan. Ramverket visar på en riktning som är intressant.

NCWIT (2009) beskriver att helst ska undervisningen ske i par framför en dator. De gör att de kan förklara, formalisera och komplettera sin tankar med sin kamrat.

Som nämnts tidigare är det viktigt att läraren styr på rätt sätt (Mayer, 2004). Det behöver vara en styrning men den får inte var för strikt. Samtidigt får eleverna inte släppas för fria då de tappar målet.

För att en transfereffekt ska vara möjlig så behöver det finnas en närhet mellan originaluppgiften och måluppgiften. Hur denna närhet kan se ut har redovisats i avsnitt 0

Transfer.

I en studie visade det sig att visualisering användes av lärare för att illustrera olika koncept i programmering (Benton et al., 2017). Genom att använda flera olika typer av representationer (en del var inte visuella) blev det lättare för eleven att förstå.

3.10. Programmering och lärandeteorier

Genom tiderna har tankarna kring hur vi lär förändrats (Greeno, Collins, & Resnick, 1996). Dessa tankar har även genomsyrat tankarna på hur vi lär av och med programmering. Före Papert (1984) var det ett behavioristiskt/empiristiskt perspektiv som gällde. Eleven skulle skapa program från tydliga mål (Mayer, 2004). I princip skulle eleven kunna arbeta själv och genom att stegvis konstruera. Deras kunskap skulle upptäckas fram. De gjorde något, fick återkoppling på det, genom att det inte blev som de tänkte kunde de lista ut vad de skulle göra härnäst.

> Students need enough freedom to become cognitively active in the process of sense making, and students need enough guidance so that their cognitive activity results in the construction of useful knowledge. Various forms of guided discovery seem to be best suited to meet these two criteria. (Mayer, 2004, sid. 16)

Papert (1984) som har ett kognitiv/rationalistisk perspektiv. Perspektivet är även beskrivet i Greeno et al. (1996). De betonar att det är viktigt att eleven ska förstå koncept och principer. Förståelsen utvecklas genom att eleven genom små steg utvecklar sina kognitiva strukturer. Även om detta rimmar väl med tankarna kring errorless learning (Greeno et al., 1996) (som ett behavioristiskt perspektiv). Metoden strävar efter att ge små belöningar (klara uppgifter) och undvika bestraffningar (att få felmeddelanden), den har visat sig att inte fungera så bra. Konversationen är också viktig främst med läraren (Mayer, 2004) att sitta själv och lära sig programmering fungerade inte då och inte nu heller.

Datalogiskt tänkande tex. Wing (2006) ställer sig till situerade/pragmatiska sättet (Greeno et al., 1996) att se på lärande med programmering. Det innebär att läraren blir central för i sin roll som mentor för gruppen. Att dela tankar och idéer inte bara genom programmeringen utan också inom gruppen blir viktigt. Dock är det så att många övningar som finns kring programmering ställer sig till det behavioristiska/empiristiska perspektivet. Ett exempel är hour of code[5] där eleverna självständigt ska lösa små uppgifter som sedan gradvis blir svårare. När de är klara med respektive övning får de återkoppling. Ben-Ari (1998) beskrev att konstruktivistiska inlärningsteorier (koginitiv/rationalistik) tillämpade på datavetenskap betonar att eleven själv skapar och är aktiv i sin inlärningsprocess. De behöver med andra ord vara aktiva i problemlösningsprocessen och hur de tänker kring programmeringen. Det konstruktivistiska synsättet vurmar även för att kunna arbeta med fysiska objekt, att handgripligen kunna ta på saker. Det blir konkret när robotar används vid programmering (Flannery & Bers, 2013; Papert, 1984).

Annan forskning har framhävt vikten av att skapa en verklighetsnära kontext för programmeringsundervisning. Om den kan relateras till elevernas intressen och förståelse, ökar det värdet av att en rik diskurs skapas kring begreppen i programmering (Grover & Pea, 2013). För att en så kallad transfer enligt det kognitiv/rationalistisk

[5] https://hourofcode.com/se Projekt för att få elever att prova på att programmera

perspektivet ska kunna ske från tex. en programmeringsövning till andra situationer måste eleven förstå den bakomliggande metoden (Greeno et al., 1996). Men det är väldigt beroende på situationen om den kopplingen kan göras. Det är lättare om de förstår datalogiskt tänkande[6].

I 3 Teori avsnittet har en diskussion först kring programmeringens/datalogiskt tänkandets bidrag till andra ämnen. Redan tidiga studier visade att det fanns tankar att programmering kunde ge ett lärande inom andra områden, dock har detta kritiseras en hel del. Betydelsen av närhet mot det område som ska läras är viktigt. För att kunna se vad programmering kan bidra med behöver man veta vad det är. En distinktion har beskrivits i uppsatsen mellan programmering och datalogiskt tänkande. Programmeringen syftar till att konkret jobba med program. Medan datalogiskt tänkande är svårare att få grepp om. Oftast beskrivs det med en mängd olika begrepp som gör den allmängiltig, allt passar in. Problemet blir då att det blir svårare att använda den och särskilja vad som hör hemma. Samtidigt beskrivs datalogiskt tänkande relativt lika mellan olika artiklar. De viktigaste delarna är abstraktion, generalisering, nedbrytning, algoritmer och felsökning (Angeli et al., 2016). I slutet har en koppling gjorts mot undervisningens kontext.

[6] jmf procedurell kunskap

4. Metod

I uppsatsen ska lärares uppfattningar om programmering beskrivas. Uppfattningarna beskrivs med både kvalitativ och kvantitativ metoder. Upplägget i denna uppsats tar ett avstamp i en genomgång av litteratur. För att komma åt de kvalitativa värdena valdes en fokusgrupp som metod. Fokusgrupp låter deltagarna samarbeta och inspireras, samtidigt som de ska förhålla sig till ett material. Fokusgruppens perspektiv har sedan använts för att skapa en enkät. En enkät är oftast kvantitativt till sin karaktär, det innebär att båda perspektiven kan beskrivas. Samtidigt kan enkäten användas mer kvalitativt genom att ha flera öppna frågor, vilket har gjorts i detta fallet. Enkäten har sedan följts upp med en intervju, detta för att säkerställa att resultaten har tolkats korrekt. Denna empiri kommer sedan att förstås med hjälp av teorin för att beskriva hur programmeringsövningar bidrag till lärande kan struktureras.

4.1. Forskningsetiska överväganden

Vetenskapsrådet (2002) har beskrivit fyra forskningsetiska krav när det gäller forskning inom humanistiska och samhällsvetenskapliga ämnen. Dessa krav är (1) informationskravet, (2) samtyckeskravet, (3) konfidentialitetskravet och (4) nyttjandekravet. I samband med att lärarna bjöds in fick de information (1) om syftet med undersökningen. Även i samband med fokusgruppens arbete och enkäten, så informerades de om undersökningens intention. Eftersom de själva valt att delta är det frivilligt (2), de har inte fått någon ersättning för deltagandet. Datainformationen förvaras på ett sådant sätt att risk för otillbörlig spridning (3) av informationen är låg. Resultatet redovisas också på ett sådant sätt att ingen person ska kunna identifieras. Det är därför intervjun redovisas under fokusgruppen (4). Resultatet kommer ej heller att användas i andra ändamål än det som respondenterna informerats om.

4.2. Fokusgrupp

Fokusgrupp kan förklaras som en strukturerad gruppintervju/diskussion. Där deltagarna genom att diskutera ett område kan lyfta varandra och ge nya perspektiv. I bakgrunden finns Vygotskys tankar där individer utvecklar varandra genom en interaktion (Greeno et al., 1996). Dynamiken i gruppen är viktigt att ta hänsyn till, då det är det som skiljer fokusgruppen från gruppintervju och informellt gruppsamtal (Wibeck, 2010). Dynamiken uppstår genom att det blir en diskussion mellan deltagarna. En skillnad mot de andra två metoderna: gruppintervju är interaktionen främst mellan moderatorn och deltagarna, i gruppsamtal är interaktionen mer informell, oftast har man ingen artefakt som deltagarna ska förhålla sig till.

Krueger och Casey (2015) skriver att en liten fokusgrupp ger en bättre förståelse av ämnet. Om personerna är kunniga, erfarna och det är komplexa frågor som ska ställas är också en liten grupp bättre. Det finns olika åsikter hur stor en grupp ska vara. Mellan tre och 12 personer är lämpligt enligt Wibeck (2010).

Lärarna till fokusgruppen rekryterades genom ett IKT-lärarnätverk RUC-IT[7] genom en förfrågan via mejl[8]. Andra sätt att rekrytera försökspersoner var att alla lärare på två skolor kontaktades, samt en förfrågan skickades till AV-media i Kronoberg som har kontakt med lärare samt ett matematik nätverk. Dessa gav dock inga fler personer till fokusgruppen. Det var frivilligt att delta och 2 personer nappade till slut på erbjudandet. Enligt Wibeck (2010) är 2 personer för lite för att använda till en fokusgrupp. Tanken var att det skulle vara flera, från början var det 6 st som anmält sig. Med kort varsel var

[7] http://rucit.net/ ett nätverk för lärare mot IKT frågor

[8] Se bilaga 1 för brevets utformning

det 4 st som inte kunde komma till mötet. Eftersom uppsatsen även har enkät och en intervju som datainsamlingmetoder, kompletterar metoderna varandra.
En bakgrundspresentation av deltagarna kommer att göras i 5.1 Diskussion kring programmeringsövningar.

Detta är inte ett helt slumpmässigt förfarande, vilket kan påverka resultatet. Det får tas med vid analyserade av data från uppsatsen. En vecka innan fokusgruppen skulle hållas skickades en detaljerad beskrivning om tillvägagångsätt, syfte, en vägbeskrivning ut samt en enkät med bakgrundsfrågor.

Frågor som ställs i en fokusgrupp har olika syften. Dem ska låta deltagarna bekanta sig med varandra. Det är bra om dessa frågor vid inledningen handlar om fakta och inte attityder, så att de kan känna en gemenskap. De fick även lite fika, Krueger och Casey (2015) skriver även att mat är ett bra sätt att stimulera gruppens diskussion. Deltagarna fick reflektera lite friare mot programmering innan ämnet introducerades. De fick sedan titta på en demonstration av de olika övningarna[9]. Under övningarna fick de reflektera över tre frågor:

- Vad lär sig eleven i övningen?
- Hur kan det som lärs ut via programmeringsövningen bidra till att nå kunskapskraven/centrala innehållet i ditt ämne?
- Vilka möjligheter och hinder ser ni för att införa och använda programmeringsövningen som en del i er undervisning?

Det är samma frågor som finns i uppsatsens problemformulering. Anledningen är att det genom detta finns en tydlig koppling mot syftet. De fick även summera sina tankar kring programmering och övningarna i slutet på fokusgruppen.

Mötet spelades in med ett antal olika konferensmikrofoner via Adobe connect. Även en lös ljudinspelnings utrustning användes för backup av dokumentationen av fokusgruppen, Zoom H5. Parallellt fördes anteckningar på papper för att fånga upp tankar som inte tydligt kanske kom med i det inspelade materialet. Därefter genomlyssnades materialet flera gånger och valda delar av materialet transkriberades.

Vid nästan all kvantitativ forskning görs ett urval som det sedan dras slutsatser från, för att sedan kunna generaliseras ifrån. Fokusgruppen däremot är en kvalitativ metod, vilket innebär att metoden används för att hitta nyanser i materialet snarare än att testa existerande idéer (Krueger & Casey, 2015). I studier som Lye och Koh (2014) efterlyser de mer kvalitativa studier när skolan och programmering studeras.

4.3. Val av övningar till fokusgruppen

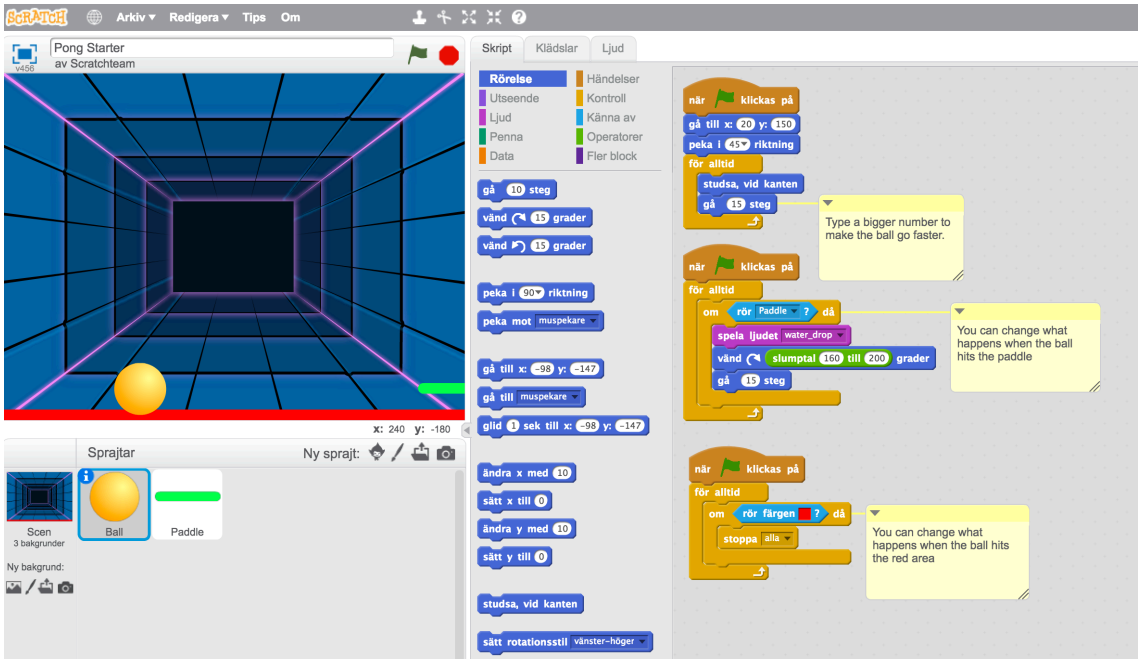
När en lärare vill börja med programmering är det troligt att de går ut på Internet för att leta reda på lämpliga resurser. Anledningen till detta är att det idag inte finns väl utvecklade läromedel annat än till gymnasiets programmeringskurser. De möts då av resurser som kan vara tex. blogginlägg om hur andra har arbetat eller tänkt kring programmering.

[9] se bilaga 5 för övningsbank och bilaga 2 för frågor som ställdes

Men det kan också vara rena programmeringsövningar som olika organisationer har lagt ut, vilket i praktiken innebär lektionsförslag (övningar). När det till denna uppsats söks efter övningar har det grovt utkristalliserats tre kategorier: (1) Att eleven med små steg utvecklar olika program. (2) De arbetar mot någon del av datalogisk tänkande.(3) Det är färdiga program som eleven skriver av eller modifierar. De är dock få av dessa övningar som direkt kan knytas till ett annat ämne. De är fokuserade på att lära ut just programmering. Exempel på ställen där det går att hitta programmeringsövningar av denna typ kan vara:

- https://www.kodboken.se/
- https://codemonkey.nok.se/challenges/1
- https://code.org/
- https://www.w3schools.com/
- https://www.cs-first.com/materials
- http://geekgirlmini.se/category/aktiviteter/
- http://scratched.gse.harvard.edu/ct/assessing.html
- https://lab.wolframcloud.com/app/
- http://microbit.org/

Vid en subjektiv analys kan man säga att dessa övningar är riktade mot olika nivåer. En del är för förskoleåldern medan andra fungerar upp till gymnasiet. En del använder textbaserad programmering medan andra använder blockprogrammering. Blockprogrammering innebär att kommandon i form av ett grafiska objekt som dras in till sin plats (figur 2). Varje block representerar ett programmeringskommando. Block/kommandon som fungerar tillsammans, passar även ihop (som pusselbitar).



Figur 2 Exempel på ett Pong-spel som styrs med musen från https://scratch.mit.edu/projects/10128515/

För att ta fram övningar till fokusgruppen har sökningar på Internet gjorts där en mängd bloggar och webbsidor av typen ovan studerats. Dessa har gåtts igenom för att hitta exempelövningar som valts ut rakt av, modifierats eller inspirerat till att utveckla helt nya övningar.

Övningarna har sedan lärarna i fokusgruppen fått kommentera och diskutera. Kriterierna för att välja ut/skapa övningarna har varit:

- Det ska vara olika svårighetsnivåer på övningarna.
- De ska kunna kopplas mot ett annat ämne.
- De ska representera både programmering och datalogiskt tänkande.
- Det ska vara olika typer av övningar ifråga om.
    - Sätt att implementera kod.
    - Resultatet av programmeringen.
    - Individuella eller samarbetsövningar.

Övningarna kan hittas i Bilaga 5.

4.4. Enkät

Enkätens frågor kan hittas i bilaga 4. Sentance och Csizmadia (2016) har bidragit med frågor kring lärarnas perspektiv, de beskriver i sin studie strategier hur lärarna hanterar utmaningarna att undervisa i programmering. Pea och Kurland (1984) har beskrivit en modell hur en programmeras kunskapsnivå kan speglas. Cohen, Manion, och Morrison (2011) som beskriver enkätutveckling ur ett allmänt perspektiv. Dessa tre källor tillsammans med tankar från fokusgruppen har legat för grund för enkäten. Enkäten har genomförts med verktyget Google forms, som är ett webbaserat enkätverktyg. Enkätens 17 frågor har strukturerats efter

- Åsikter om programmering i allmänna ordalag (4 st)
- Om de använt sig av programmering (1st)
- Hur undervisning kan se ut? (5st)
- Deras egen uppskattade kunskapsnivå (4st)
- Bakgrundsinformation (4st)

Enkäten har distribuerats genom sociala medier[10]. Twitter var en kanal och Facebook var en annan. På Facebook har inlägg gjorts i flera grupper som har med matematik, IKT och skola att göra. Eftersom respondenterna kan ha interagerat med inläggen sker ingen närmare beskrivning än så pga. konfidentialitetskravet (Vetenskapsrådet, 2002). Det är svårt att räkna ut svarsfrekvensen när enkäten distribuerats via sociala medier. Teoretiskt har den nått ut till 20 000 personer. Dock är inte alla lärare med på Facebook, det inte alla som ser meddelandet, ännu färre som agerar och fyller i enkäten. Risken finns även att folk som inte är lärare fyller i enkäten. Till viss del kan frågornas natur hindra det. Tex kan frågorna röra sådant som bara en lärare kan svara på. Som jämförelse vid utskick via e-post vet man exakt hur många enkäter som sänts ut och kan därefter räkna ut svarsfrekvensen. Så är inte fallet vid sociala medier som kanal, tex. börjar man lägga ut påminnelser så kan det räknas som spam. I värsta fall kan man bli avstängd från grupperna. Det blev 74 personer som svarade på enkäten. En bakgrundspresentation av deltagarna kommer att göras i resultatavsnittet.

Enkäten har utformats med flera öppna frågor vilket gör att den är lite mer kvalitativ än en enkät med flervalsfrågor eller likertskalor[11]. Anledning är att det är svårare att få gångbara resultat från kvantitativa frågor då urvalets position är osäkert (Mayring, 2007). Givetvis kommer även det kvantitativa resultaten att redovisas där det är lämpligt.

[10] Se bilaga 2 för utformandet av inlägg

[11] https://sv.wikipedia.org/wiki/Likertskala Rangordningsskala som används i enkäter

En fråga handlade om de hade undervisat i programmering. Denna fråga delade upp urvalet i 2 grupper: De som inte provat och de som mer eller mindre provat att undervisa i programmering. De kommer i uppsatsen benämnas Grupp 1 och Grupp 2. Detta för att se om dessa två grupper har olika uppfattningar om programmering. En hypotes i uppsatsen är att de som har prövat att använda programmering antagligen har tänkt mera kring innehåll och syfte med undervisning i programmering.

Bakgrundsenkäten till fokusgruppen och enkäten till sociala medie-grupperna har en liknade utformning. Eftersom fokusgruppens enkät skickades ut först har den legat till grund för sociala medie-enkäten. Det är så pass stora likheter mellan dessa, så fokusgruppens enkät har inte redovisats.

4.5. Intervju

För att undersöka relevansen av de tidigare observationerna från fokusgrupp och enkäten har en intervju genomförts. Den är baserad på upplägget från fokusgruppen. Övningarna förevisades och den som intervjuades fick kommentera dem. Demonstrationen/diskussionen följdes upp med ett samtal som var baserat på data från svar som lämnats av fokusgruppen och enkäten. Tanken med intervjun är att stämma av de resultat som kommit fram vid fokusgruppen och enkäten. Det har hållits en intervju med en lärare som tog ca 1 timme. Intervjun redovisas som en del i fokusgruppen.

4.6. Avgränsningar

I detta avsnitt beskrivs ett antal avgränsningar som fokuserar studien.

En sökning i universitetets artikelkatalog Onesearch[12] gav drygt 52000 träffar på ordet ”computational thinking”. Det innebär att området är för stort för att läsa alla artiklar. Processen har därför inriktat sig på att läsa några texter som har varit ofta citerade. Detta har kompletterats med granskande artiklar som diskuterar dessa texter.

För att kunna begränsa materialet har 8 övningar valts ut som lärarna ska ha åsikter om vid fokusgruppen. Fler övningar hade gett en mer nyanserad bild över området.

Läroplanen i skolan reglerar undervisningen (Skolverket, 2016a). Det finns olika nivåer i läroplanen som läraren måste ta hänsyn till, som till slut konkretiseras i kursplanen för respektive ämne. I läroplanenn finns övergripande mål, vilket leder till kursplanen i ämnet som har syfte i ämnet, förmågor i ämnet, centralt innehåll och kunskapskraven. I de övergripande målen står det bl.a.

- kan lösa problem och omsätta idéer i handling på ett kreativt sätt,
- kan lära, utforska och arbeta både självständigt och tillsammans med andra och känna tillit till sin egen förmåga. (Skolverket, 2016a)

I ”syftet” i ämnet, ”förmågor” i ämnet, ”centralt innehåll” sätts ramarna för undervisningen upp. Dessa ramar är viktiga och ska beaktas vid undervisning. Det är dock utifrån ”kunskapskraven” som läraren ska bedöma eleven, vilket gör dem extra viktiga. I denna uppsats definieras lärandet i ämnet som att sträva efter att nå kunskapskraven och sedan förhålla sig till de centrala innehållet.

[12] https://lnu.se/ub/

I frågeställningen beskrivs ”andra ämnen”, med detta syftar i denna uppsats på de ingående lärarnas (från enkät och fokusgrupp) ämnen och på de ingående övningarnas inriktning. Med andra ord är inte matematik och teknikämnet med som annars kan vara naturligt att ha med.

En del övningar är knutna till vissa plattformar för att fungera. Infrastrukturen kan för lärarens val ha stor betydelse tex. om de bara har datorer, fungerar inte appar. I detta arbete har ingen hänsyn tagits till det utan diskussionen har förts generellt.

Övningarna som beskrivs finns givetvis i en kontext. Denna kontext kan påverka hur övningen fungerar. I denna undersökning har kontexten begränsats till lärarnas svar och den forskning som presenteras.

4.7. Forskningsetiska överväganden

Vetenskapsrådet (2002) har beskrivit fyra forskningsetiska krav när det gäller forskning inom humanistiska och samhällsvetenskapliga ämnen. Dessa krav är (1)informationskravet, (2)samtyckeskravet, (3)konfidentialitetskravet och (4)nyttjandekravet. I samband med att lärarna bjöds in fick de information (1) om syftet med undersökningen. Även i samband med fokusgruppens arbete och enkäten, så informerades de om undersökningens intention. Eftersom de själva valt att delta är det frivilligt (2), de har inte fått någon ersättning för deltagandet. Datainformationen förvars på ett sådant sätt att risk för otillbörlig spridning (3) av informationen är låg. Resultatet redovisas också på ett sådant sätt att ingen person ska kunna identifieras. Det är därför intervjun redovisas under fokusgruppen (4). Resultatet kommer ej heller att användas i andra ändamål än det som försökspersonerna informerats om.

5. Resultat

Detta avsnitt inleds med en redovisning av fokusgruppens bakgrund och åsikter om programmering. Därefter analyseras övningarna tillsammans med fokusgruppens lärares reflektioner. Därpå redovisas enkätens resultat i två delar dels ett avsnitt som beskriver deras bakgrund och förutsättningar för att undervisa och dels deras tankar kring undervisning i programmering.

5.1. Diskussion kring programmeringsövningar

I fokusgruppen + intervju har 2+1 lärare diskuterat åtta programmeringsövningar och diskuterat om programmering kan bli ett verktyg för lärande. Lärarna i fokusgruppen har en gedigen bakgrund i programmering, då båda två läst datavetenskap. Ena läraren undervisade på gymnasiet och andra på mellanstadiet. Den intervjuade läraren undervisade i andra ämnen matematik och programmering på högstadiet och hade ringa erfarenhet av programmering.

Eftersom programmering inte har ingått i läroplanen annat än till kursen Programmering på gymnasiets tekniska program, har de inte arbetat så mycket med programmering i andra ämnen. Det de har använt är tex. C#, Ardurino programmering, men även Scratch i form av att skapa en berättelse och hour of code.

De ser möjligheter att programmera visualiseringar i NO samt i matematiken för att lösa uppgifter. Mellanstadieläraren hade även tankar om att arbeta med det i svenska genom berättelser. En generell analys av deras kommentarer av övningarna, gav att de tog upp olika datalogiska begrepp, när de beskrev vad man lärde sig av övningarna. Det trots att jag ställt frågor där jag efterlyste andra delar som kunskaper och färdigheter från läroplanen. På någon fråga var närheten från övningen till ett annat ämne stor och då kom också andra delar fram.

En lärare poängterade att för att programmera måste kunskapen om programmering finnas. ”Om vi har 5000 mätvärden går det inte att räkna ut förhand, utan måste ta hjälp av datorn. Men där är ofta de här (brist på)[13] programmeringskunskaperna ett hinder, dom har inte tagit det här steget, de kan inte språket och då måste man börja med det och då kommer man ifrån själva problemet.”

De betonades att datorn är nödvändig och många övningar på den är indirekt programmering också, det blir ett hjälpmedel för att upptäcka. En gymnasielärare:

> Använda datorn som verktyg för att skala upp problemet till en verklig nivå, tex. där man kan räkna standardavvikelser för några få värden. Ja då kanske man kan ta något verkligt problem, man har gjort temperaturmätningar under i flera år, ska titta på medeltemperaturen ökar eller vad hände. Då måste vi ta hjälp av datorn. Det gör att man kan titta på intressantare problem.

5.2. Fokusgruppens övningar

Under detta avsnitt redovisas lärarnas uppfattning kring övningarna. Deras beskrivningar har vävts samman med intentionen med övningen, vad de kan lära mot andra ämne och problem med övningarna. Allt detta för att sätta dem i perspektiv. I 4.3

[13] Troligen en felsägning, ”brist på” borde in här

Val av övningar till fokusgruppen beskrevs vilka kriterier som användes för att välja ut övningarna, de är bra att ha i åtanke.

Övningarna finns beskrivna i bilaga 5. I detta avsnitt kommer övningarna att diskuteras. Till grund för diksussionen finns kriterierna i 4.3 Val av övningar till fokusgruppen. Den ska också reflektera kring intentionen med övningen. Vad övningen lär till andra ämnen. Samt utmaningar och möjligheter med dessa övningar. Detta är kryddat med fokusgruppens åsikter.

5.2.1. En övning mot hjärnas delar

Denna scratchövning är ett exempel där flera olika delar i undervisningen kommer in. Intentionen är att eleverna ska arbeta tillsammans vilket främjar samarbete. Det är en del problemlösning som måste göras dels i själva programmeringen och dels hur de praktiskt ska dra sladdar för att det ska fungera. Innehållet som i detta fall var hjärnas delar får de interagera med på ett annorlunda sätt. Det stämmer väl med forskning som visar att variation är viktig del i undervisningen (Ling & Marton, 2011). En lärare uttryckte det som ”De får lära sig det lite mer handgripligt att få ta på hjärnas delar och det kanske fastnar antagligen bättre än om de läser det i en bok". En annan sa ”Finns en risk att de fastnar i spelandet och tappar fokus”. Ett problem är att det inte blir så mycket programmering om de köper programmet rakt av. De behöver då bara ändra på någon variabel. Fokusgruppen menade att övningen kan fungera i praktiken för yngre barn, om mycket är förberett. Det får dock inte bli för mycket fokus på att klippa och klistra.

5.2.2. En chattrobot

Denna övning kan göras väldigt avancerad. Här kan de mycket väl lära sig naturligt språk som en del av övningen. Dels genom att programmera de regler där roboten ska svara, vilket ger en insikt hur språket fungerar. Dels genom att om de arbetar i par, kan de interagera med varandra som gör att de tvingas kommunicera. För att övningen ska fungera behöver de ha kommit en bit i sin kognitiva utveckling. Därför kan det tänkas att den fungerar bättre om ska träna engelska i åk 6 eller på SFI utbildningen än från de första åren i skolan. Givetvis beror det på exakt hur övning blir designad i sin kontext. I fokusgruppen sa en lärare ”Kan bli jättespännande, så får min kompis testa den få se om den fungerar eller svarar kompisen på ett annat sätt, det blir en diskussion kring dialogen. Behöver jag ändra något, är det något som inte är tydligt”.

5.2.3. Textprogrammering (Silent teacher)

Detta är en klassisk programmeringsövning som för tankarna till Papert (1984). Eleven bedriver upptäcktsbaserad inlärning. Men det är fel fokus på vad som ska upptäckas menar Victor (2012). Det är viktigt att meningen är transparant i verktyget. Miljön måste göra det möjligt för eleven att enkelt förstå programmet, för att avkoda koden, så att hen kan koncentrera sig på programmeringen - hur de algoritmiska "ingredienserna" kombineras. I detta fall blir det mera att gissa sig fram till rätt svar. Victor påpekar att Paperts tankar är korrekta i fråga om att övningen ska visualisera koden tex. block-programmering istället för att skriva koden. Interaktionen med text i övningen är det inget fel på och kan säkert tilltala en del. Men varför ska eleven arbeta i blindo för att sen plötsligt upptäcka det rätta svaret. Troligen är det för att de anser att de steg som svårighetsgraden ökar i är så pass små att rätt svar blir ett naturligt nästa steg. Fokusgruppen tyckte att det här introducerade matematik med ekvationer, algebra på ett bra sätt, men att eleven måste vara driven för att de ska kunna arbeta självständigt. De tyckte också att läraren måste finns till hand för introduktion och handledning.

5.2.4. En övning om digitala val

En mer diskussionsövning då eleven inte kommer i kontakt med ren kod. Den bygger istället på att tänka datalogiskt. Grundproblemet vid starten är, hur ska val gå till? Sedan lägger läraren till fler och fler parametrar vilket gör övningen mer komplex. Tanken är att eleverna genom övningen erfar hur val går till, genom att de ”tvingas” att hantera kod. I kursplanen i samhällskunskap finns det inskrivet att man ska arbeta med demokrati. En lärare kommenterade: ”Bra upplägg datalogiskt eftersom man börjar med ett problem, som ska lösas, sedan kommer man till nästa problem, blir att man delar upp det.” Övningen ger även en introduktion till grundläggande datasäkerhet tyckte en lärare.

5.2.5. Gapminder

Genom att manipulera olika variabler får eleven syn på intressanta samband. De kan tex. vara att se hur stora händelser påverkade olika variabler. Programmeringen begränsas till att variera olika variabler. De intressanta sambanden kan vara svåra att upptäcka själv. Då oftast övningarna går ut på att eleverna får välja variabler utefter färdiga mallar, men även spel och frågesport finns i lektionsmaterial arsenalen (Friberger, 2015). En lärare kommenterade:

> Man tittar på beroende av olika variabler. Det är tiden som är variabeln, det kan vara lite förvirrande i början. Man är ju van med att den oberoende variabeln på x-axeln och den beroende på y-axeln. Men här finns en tredje som är tiden.

5.2.6. Googlesökning

Logik är en viktig del i programmering. Ett sätt att förstå det är att förfina sökningar på Google. Det tradtionella är att man skriver fler och specifika ord för det man är ute efter. Det finns även många andra sätt att förfina sökresultatet enligt Googles hjälpsidor[14], tex. att sätta ett minus tecken framför ett ord för att exkludera det. Problemet är att Google använder artificiell intelligens för att leverera resultatet. Det innebär att den gör kvalificerade gissningar över vad den som söker vill ha. Det är bra men inte om man vill få syn på ett tydligt samband mellan sökningen och sökresultatet. Fokusgruppen höll med om att detta är en bra introduktion till programmering som att träna logik och boolesk algebra.

5.2.7. Kompass

Micro:bit har tagit fram en stegvis tutorial[15] som låter en elev skapa en kompass. Den beskriver ingående alla steg. I princip skriver eleven av koden och får sedan ett färdigt program. Detta program laddas sedan över till Micro:biten. Micro:bit som plattformen heter har delats ut till många skolbarn i England. Tanken är att eleverna ska genom denna plattform lära sig att programmera. Det finns en hel del färdiga exempel som eleven kan skriva in och det finns även blockprogrammering som eleven kan använda för att skapa program. Lee et al. (2011) presenterade en modell Use-Modify-Create. Det känns dock som ett stort steg att gå från att använda till att ändra och skapa. Samtidigt ska man inte förringa den taktila interaktion som micro:bit kan skapa, det är säkert

[14] https://www.google.com/intl/sv_se/insidesearch/tipstricks/all.html

[15] https://www.microbit.co.uk/td/lessons/compass Enkortsdator utvecklat av BBC för utbildning av elever i datalogiskt tänkande

motiverande (Papert, 1984). I detta fall ska eleven lära sig något om hur en kompass fungerar, då grader och väderstreck är en del av programmeringen.

5.2.8. Ett arbetsplatsspel

Eftersom ”spelet” utspelar sig på en arbetsplats och uppgifterna där löses genom programmering, tror då lärarna att eleven kommer att lära sig något om arbetsplatser. Fokusgruppen menade att så inte var fallet. Det stämmer väl in med tankarna kring procedurell retorik (Bogost, 2008) då programmeringen inte används för att förstå arbetsplatsens dynamik. Det blir en parallell handling till en allt för grovhuggen bild av en arbetsplats.

5.3. Lärarnas bakgrund (från enkäten)

I detta avsnitt redovisas lärarnas bakgrund: dels med en allmän beskrivning av dem och dels med fokus på programmering. Detta för att det ska vara tydligt hur de förhåller sig till programmering. Som beskrivits var det 74 personer som skrev enkäten, dock var det inte alla som svarade på alla frågor. De flesta av lärarna undervisar i matematik. 96% av respondenterna har det som ett av sina ämnen (Diagram 1). Det innebär att programmering kommer att vara en del av deras undervisning i framtiden. Samtidigt undervisade de i andra ämnen, en hypotes är att de tar med sig tankar om programmering även till dessa ämnen.

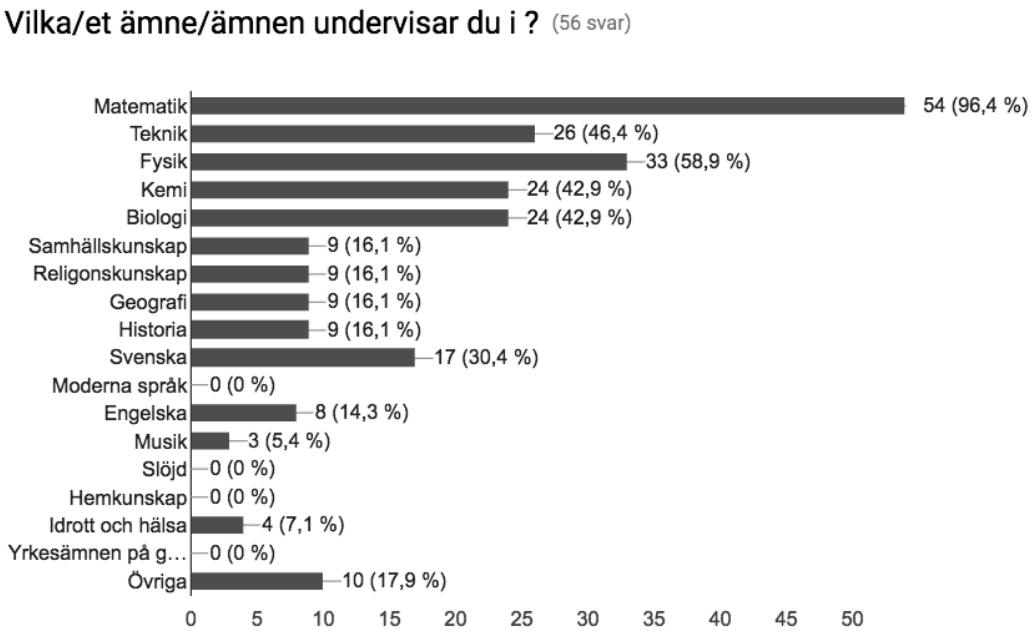


Diagram 1 Vilka ämnen som de undervisar i (de kunde kryssa i mer än ett ämne)

Lärarna var inriktade mot att undervisa med en tonvikt på de äldre stadierna (högstadium och gymnasieskolan), men det var relativt jämnt fördelat. Det innebär att programmeringsundervisningen kan och ska vara något mer avancerad än om den vänder sig till yngre elever. Det vanligaste åldersspannet var mellan 41-50 år med en jämn fördelning av både yngre och äldre.

Av de 62 st som besvarade frågan om vilken nivå de var på var det 42 st som bedömde på en nivå programanvändare (diagram 2). Vilket är den lägsta kunskapsnivån.

Resterande var jämt fördelade på de övriga alternativen. Klassificering är hämtad från Pea och Kurland (1984)

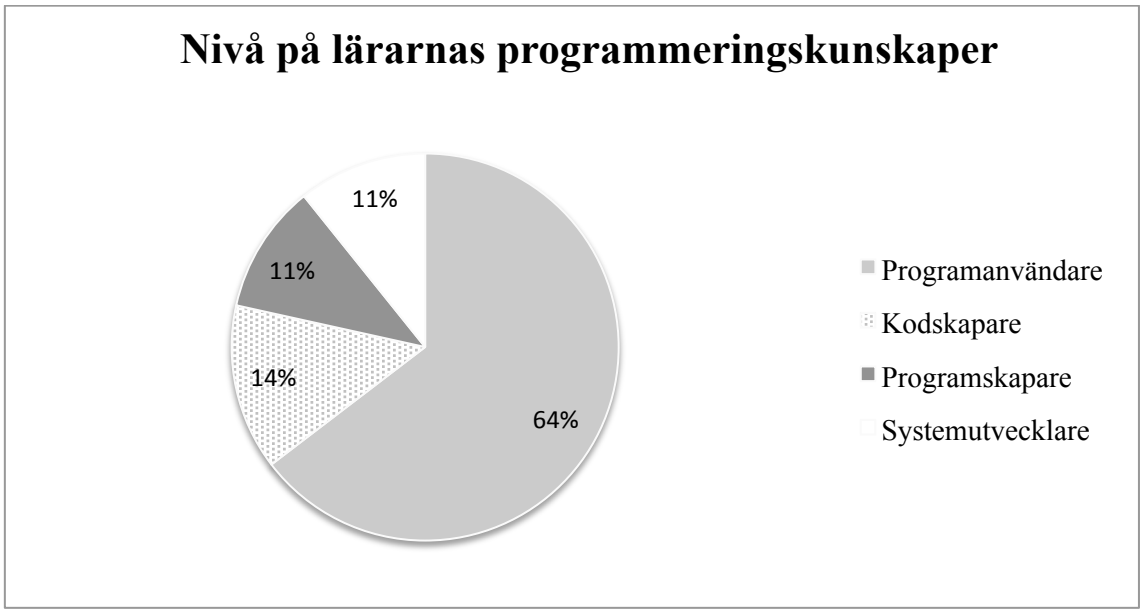


Diagram 2 Självskattning av lärarnas programmeringskunskaper

28 % av respondenterna kunde inte programmera när de fick bedöma sig själva (Diagram 3). Av de övriga som kunde programmera, var många autodidaktiker. En del hade även gått datavetenskapliga kurser på universitet. Några få nämnde andra aktörer som AV-media som kunskapskälla. Lärarutbildningens bidrag var det få som tog upp, vilket inte är så konstigt då det inte legat i deras uppdrag. Bland de övriga nämndes kollegor och familj som alternativ. Eftersom de kunde kryssa i flera alternativ är antalet val större än antalet deltagare, ungefär varannan person har kryssat i ett extra alternativ. Några av dessa har även kryssat i ett tredje alternativ.

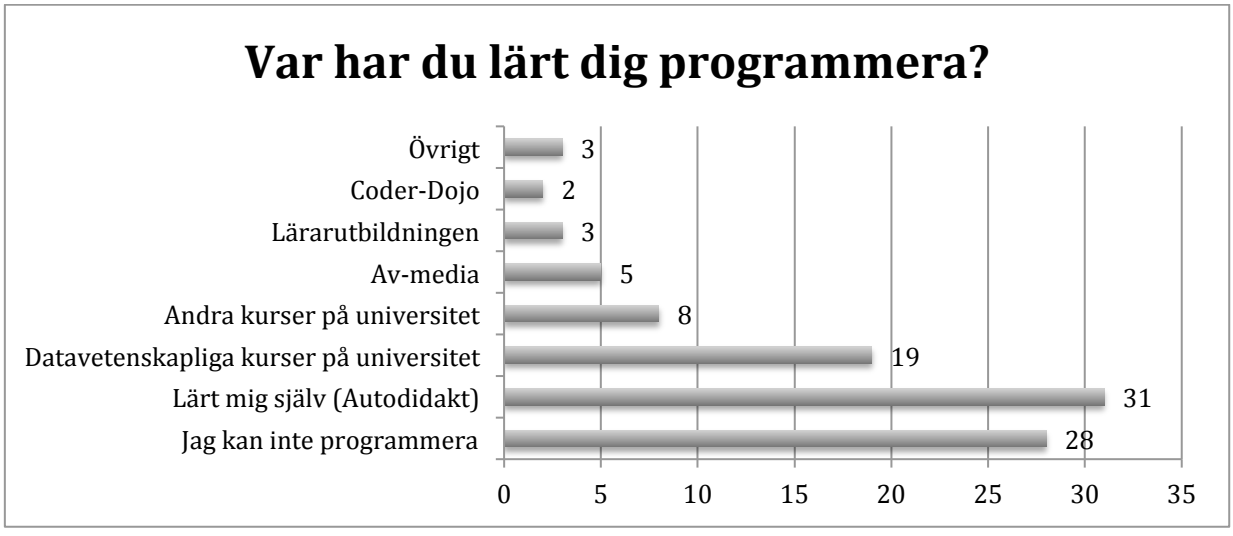


Diagram 3 Kunskapskällor

På frågan vad de lärare som inte undervisat i programmering behöver för utbildning kring programmering (Grupp 1), var det många som svarade ”Allt”. Annars efterlyste de kunskaper om olika programspråk. Flera tog också upp didaktiska perspektiv och hur undervisning av programmering ska gå till. Det var ett vanligare svar bland de som har provat att undervisa i programmering (Grupp 2) än i Grupp 1. Bland de som inte testat att undervisa i programmering (Grupp 1) ville de lära sig att koda, vilket är en

instrumentell förmåga. Resultaten visar också att det var 45% som inte använt programmering i sin undervisning det senaste året (Grupp 1). Det innebär samtidigt att det är 55% som provat (Grupp 2). Grupp 2 tyckte även att fokus ska ligga på att praktiskt tillämpa programmering mer än en fördjupning i teorin.

5.4. Lärarnas åsikter om undervisning i programmering

Här redovisas lärarnas åsikter från enkäten. Lärarna är de som ska bedriva undervisningen. Det innebär att 38% rankade (1) sig som otrygga (Diagram 4) när det gäller att undervisa om programmering. Totalt var det är en övervikt på de som är på den otrygga halvan. Ett samband som finns, är att de som har en utbildning inom datavetenskap rankar sig generellt som tryggare än de som inte har det.

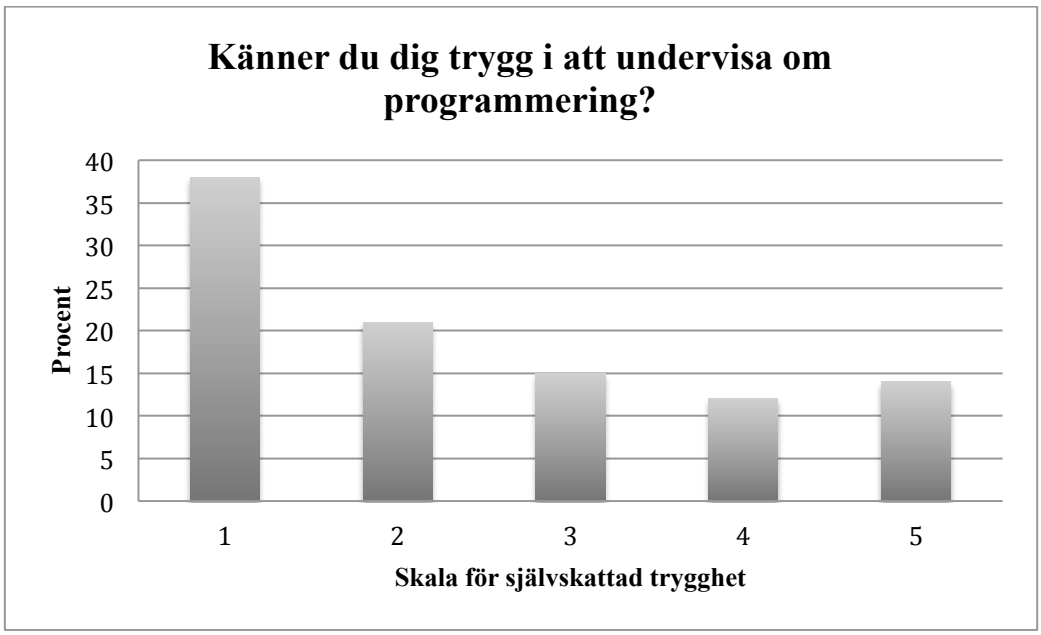


Diagram 4 Självskattad trygghet

En fråga som ställdes var om de tyckte att det var viktigt att undervisa om programmering i skolan (Diagram 5). Frågan var utformad med 4 alternativ. Det gjorde att de var tvungna att välja. Resultaten visar att det är fler som tycker det är viktigt 62% än de som inte gör det (38%).

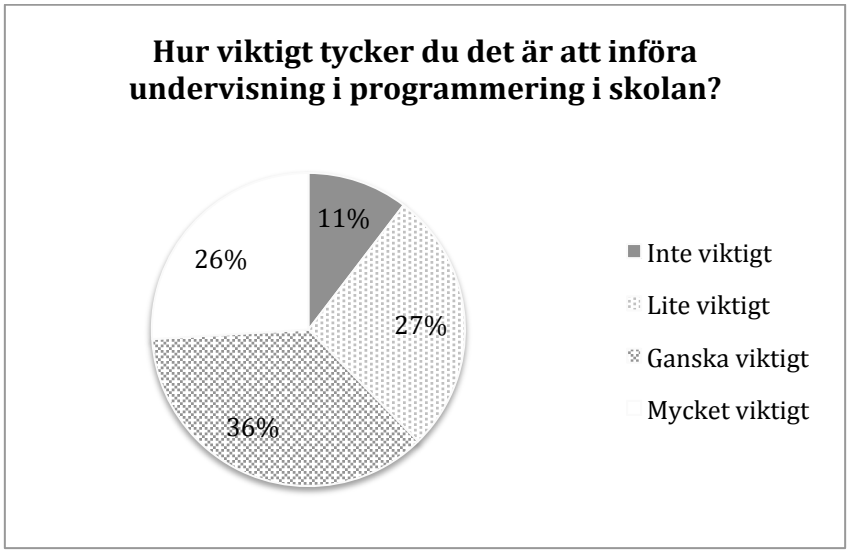


Diagram 5 Relevans för skolan

Bland Grupp 1 låg utmaningen i att få utbildning och tid till att lära sig programmera. Flera beskriver ett stort behov av relativt mycket programmeringskunskaper, för att kunna undervisa. De har även svårt att se hur det kan passa in i deras undervisning. Grupp 2 tog upp hur undervisningen ska se ut. Det var allt från att eleverna är en hetrogengrupp, som det är svårt att anpassa undervisningen till. Till att det saknades bra material samt vad som fungerar i klassrummet. Flera nämnde också att infrastrukturen måste fungera. De såg också problem med att kollegor inte var med på tåget. Även Grupp 2 tog upp tid och utbildning som utmaningar.

När Grupp 1 fick beskriva möjligheter: såg de till sin egen utveckling, träna logiskt tänkande, mer motiverade elever, mer varierad undervisning. Grupp 2 tog upp liknade åsikter. De var också inne på att det vidare perspektivet, där de tog upp nyttan för samhället, kreativitet och samarbete. Flera tog även upp kopplingen mot matematiken, att det blir lättare att förstå den. Samtidigt såg flera lärare programmering som en liten del av undervisningen, inget som kommer uppta all tid.

> Jag får intresserade och engagerade elever som undersöker tillsammans med mig som lärare. Då datorer och ipads är få till antalet får vi tänka om och arbeta annorlunda, ett led i ett logiskt och kreativt lärande. Här ser eleverna att vi skapar egna lösningar och blir producenter istället för konsumenter och det är väl en viktig del i programmering.

På frågan om vad eleverna får ut av att undervisas i programmering beskrev många i Grupp 1 att det inte är så mycket. De har ju inte undervisat så det är antagligen inte så lätt att ha en idé om det. Samtidigt visar enkäten att de inte ser möjligheten. I Grupp 2 lyfte de kunskapen om begrepp från definitionen av datalogiska tänkande: problemlösning, logik, strategi, mönster, algoritmer, struktur är begrepp som kan karakterisera deras svar. Några svar var mer utblickande med betoning på förståelse för sin omvärld och hur de praktiskt kan tillämpa sin kunskap och förhoppningsvis kan bli anställningsbara.

Grupp 1 hade få förslag på frågan "Vilka bra tekniker/strategier/övningar tror du kan hjälpa eleverna att förstå programmering?" Några förslag var dock formelhantering, primtalsfaktorisering, BeeBots. Grupp 2 speciellt de som undervisat mycket lyfte fram: "Koncept och principer framför syntax och programspråk". Samtidigt tyckte de att undervisningen behöver kryddas med lite roligare övningar från tex. code.org, kodcentrum och koda.nu. Dessa övningar ska gärna ha något spelinslag. Någon tog även upp Excel[16], där eleverna kan skriva formler, även GeoGebra[17] nämndes.

När programmering skulle beskrivas visste flera av lärarna i grupp 1 inte vad de skulle skriva. Några svarade saker som algoritmer, logik och förstå mönster. Bland de som tillhör grupp 2 beskrev datalogiskt tänkande mer nyanserat och djupare. En lärare sa "Planering, utförande, utvärdering av det skapade" en annan sa " Behärska de grundläggande koncepten (sekvens, alternativ repetition), Möjlighet att gå från problemformulering till kod via strategi, Felsökning".

[16] https://products.office.com/sv-se/excel Ett Kalkyl program
[17] https://www.geogebra.org/ GeoGebra är en dynamisk och interaktiv matematikmiljö för alla undervisningsnivåer som samlar geometri, algebra, kalkylblad, grafritning, statistik och analys i ett lättanvänt paket.

När grupp 2 beskrev programmeringens relevans för andra ämnen i läroplanen tog de upp problemlösning, samarbete, allmän kunskap om digitala verktyg, strukturera arbetet. En respondent svarade.

> Vi använder oss av progr. i så gott som de flesta ämnen. Vi tränar engelska glosor med hjälp av Bluebot, vi tränar rim och stavelser med hjälp av våra robotar. Värdegrunden kommer smidigt in genom samarbete och genom att upptäcka att vi alls kan olika saker och kan bidra med olika saker.

De är få som lyfter centralt innehåll. En lärare skrev ”… elever har programmerat spel för att visa på vikten att hålla haven rena” vilket är del av geografiämnet. En respondent nämnde att: ”Man kan t ex programmera en saga istället för att alltid skriva, man kan ändå se om eleven ser strukturen för en saga och har en röd tråd osv.” Några nämnde att det fanns sätt men inte hur de konkret kan beskrivas. Grupp 1 hade inga konkreta idéer.

6. Diskussion

Diskussionen består av fyra delar, i den första diskuteras lärarnas perspektiv, utifrån vilka utmaningar och möjligheter programmering har. I nästa del reflekteras det hur lärandet med programmering kan se ut. Sedan presenteras en modell över hur programmeringsövningar kan klassificeras. Diskussionen avslutas med en metoddiskussion.

6.1. Lärarnas perspektiv

I detta avsnitt diskuteras vilka möjligheter och utmaningar lärarna ser när det gäller att införa och använda programmering som en del i sin undervisning.

Enkäten visade att bland de lärare som inte använt programmering i sin undervisning var osäkerheten stor. De var mer negativt inställda till att införa programmering i läroplanen, såg även flera problem med införandet. Detta är dock en grupp som inom en snar framtid behöver införliva programmering i sin undervisning. De har en uppfattning om att programmering är något stort som måste införas. Någon nämnde att de behövde 3 terminer på universitet för att det ska fungera. Yadav, Mayfield, Zhou, Hambrusch, och Korb (2014) visar i sin studie att med relativt små insatser kan lärarstudenter få kunskap om datalogiskt tänkande. Dock är det säkert att behöver de skriva program, så behöver de få in kunskap i syntaxen, vilket tar längre tid att lära sig. Det visar att för att lärarna ska kunna undervisa i programmering behöver det göras en insats både i tid och utbildning.

Engelsen (2006) skrev i sin avhandling om implementeringen av IKT i undervisningen. Han konstaterade att det finns olika sätt för lärarna att ta till sig när ny teknik ska införas i undervisningen. De kan se IKT som ytterligare en sak som läggs på dem och ska göras. De blir då osäkra på vad de ska fokusera på, ”Vad är det som är viktigt i min undervisning?”. Han kallar detta fenomen för ”fokustrengsel”. Ett annat sätt att hantera införandet av IKT är att integrera det i sin undervisning. De arbetar med samma saker men får med IKT som en del av detta. En metafor kan illustrera tankesättet: om du har 1 dl socker och 1 liter vatten. Så är mängden 1,1 liter men blandar du ihop ingredienserna så löses sockret upp och du får lite drygt 1 liter. Sockret blir en del av lösningen. Diskussionen är likande när det gäller programmering. En lärare svarade på vilka utmaningar som fanns med införandet ”TID- läggs bara till nytt! Vad prioritera bort?” Givetvis behöver lärarna kompetensutveckling inom området och kommer då att se sätt att kunna integrera programmering i sin undervisning.

En lärare skrev ”Att undervisa elever i 30-grupp i programmering är omöjligt. 27 elever ropar om hjälp konstant. Jag vill inte, kan inte, orkar inte!” Citatet beskriver en annan utmaning hur undervisningen ska utformas. Uppenbarligen har undervisningen i detta fall gått för fort fram då inte eleverna hänger med. Forskning (Pirolli & Recker, 1994) visar att det är viktigt att eleverna ska förstå datalogiskt tänkande innan de börjar programmera. Motivation är en viktig del i allt lärande. Lärarna som uttryckte tveksamheter har antagligen mindre motivation. Det innebär att de behöver förstå nyttan med programmeringsundervisning i skolan för att kunna lära sig programmering.

I enkäten var det många som ville lära sig att skriva kod av de som tillhörde Grupp 1. Frågan är om de inte ska förstå koncepten i datalogiskt tänkande innan de börjar koda. En annan väg är att använda övningar som inte innehåller så mycket syntax programmering. I fokusgruppen användes övningen ”hjärnas delar”, vilket skulle kunna vara ett alternativ. De gör något som de egentligen inte tycker är programmering. Lee et al. (2011) beskrev sin modell Use-modify-create. I början räcker det med att använda programmet sedan ändrar eleven och anpassar programmet, innan eleven är mogen att skapa program själv.

Om eleven har verktyg/övningar måste verktygen även stödja datalogiskt tänkande. Inte bara att det är en programmeraringsmiljö (Victor, 2012), utan även stimulerar datalogiskt tänkande. Lärarna i enkäten tyckte att fokus skall ligga på praktik och inte teori. För att tillfredsställa den viljan behöver inlärningen av datalogiskt tänkande var praktiskt inriktat.

För att se på de mer positiva sidorna vad programmering kan bidra med såg lärarna i enkäten att det kan bli en mer varierad undervisning. Säljö (2005) använder begreppet medierande redskap, för att beskriva de kulturella redskap som vi använder för att kunna förstå, tolka och interagera med vår omvärld. Frågan är om programmering kan bli ett sådant redskap? En lärare kommenterade i enkäten sin syn.

> Jag tycker enkäten andas ett fundamentalt missförstånd om programmeringsundervisningen i läroplanen. Det handlar inte om att lära ut programmering för programmeringens skull, utan att använda programmering som ett pedagogiskt verktyg för att utforska, belysa och arbeta med andra delar av läroplanen. Exempelvis i matematiken så kan programmering i en grafisk miljö användas för att utforska geometriska element i ett tvådimensionellt koordinatsystem på mycket mer kraftfullt och flexibelt jämfört med att göra samma sak med papper och penna. Och som bieffekt tränas man i programmering, men det är inte det primära.

Syftet med uppsatsen är att göra det som läraren efterlyser. Att träna programmering för programmeringens skull har inte varit intentionen med uppsatsen. Läraren som skrev detta har mycket goda kunskaper av programmering. Tydligen uppfattar hen att någon nyans saknas.

Medierade verktyg tar sitt avstamp i det sociokulturella perspektivet. Speciellt Grupp 2 tog upp nyttan för samhället, kreativitet och samarbete, som viktiga delar i arbetet med programmering. Det gör att det sociokulturella perspektivet lyfts fram. Det är även denna vidare förmåga som skolverket eftersträvar i sin intention med att föra in programmering (Skolverket, 2016b).

Sammanfattningsvis är de som har kunskap om programmering mer positiva till införandet. De har också en djupare insikt hur programmering kan användas.

6.2. Lära med programmering

I detta avsnitt beskrivs hur det som lärs ut via programmeringsövningar bidrar till att skapa ett lärande i andra ämnen. Analysen är baserad på resultatet från fokusgruppen och övningarna med koppling mot teorin.

I denna uppsats har utgångspunkten varit några programmeringsövningar. Övningarna i denna uppsats är ryckta ur sitt sammanhang. Eftersom det är troligt att en sån här övning bara är en del i det de gör för att lära eleverna. När övningarna är använda på detta sätt är det en instrumentell syn på lärandet. Krasst innebär det att ”läraren” genom att använda övningen vill att eleverna lär sig ett område/ämne genom programmering. Dock är lärandet mer komplext än så vilket har beskrivits i teori delen.

Det som lärs när övningarna görs är först och främst programmering. För att få en reell effekt på andra ämnen behöver de vara väldigt tydliga mot det ämnet. Black och Wiliam (2010) menar generellt att eleverna måste få uttrycka sin förståelse i ett ämne för att de ska lära sig det. Är det bara påhängt, kommer de inte reflektera kring det, vilket leder till ett utebliven lärande som effekt. Det rimmar väl med Bogost (2008) som beskriver lärande från spel. Han tar där upp procedurell retorik som kan förklaras som att det är reglerna som ger lärandet, inte miljön. Tar vi Arbetsplatsspelet utspelar det sig på en arbetsplats men genom att det bara är miljön som beskriver platsen, så är det explicita lärandet inte så stort. Eftersom reglerna bara berör programmeringen. Det beror givetvis på vilken nivå som eleverna befinner sig på. En lärare såg tex. potential i att de repeterar enkel engelska, får ett strukturerat arbetssätt. Övningen ”Digitala val” har då bättre potential för lärande av ämnet då förutsättningarna (reglerna) förändras. Det får eleverna att börja tänka kring hur de ska lösa övningen på bästa sätt och får samtidigt en förståelse för problematiken kring val. I detta fall ger reglerna en mening till övningen. Övningen Chattroboten låter också eleverna arbeta med reglerna kring språk. De kan här träna på olika typer av konversationer, på ett beställa på ett Café, fråga om vägen. Speciellt på SFI kurser kan det vara ett sätt att arbeta.

I några av övningarna krävs det olika mycket programmeringskunskaper. I Google-sökningen eller i Gapminder[18] tilldelar eleverna variabler värden, möjligen gör en viss filtrering, men kan det räknas som programmering? Det är inte lika avancerat som att skapa kompassen i Micro:bit eller koda i Silent teacher. Det är ju så att mycket av programmeringen går mot moduler. Det illustreras på ett tänkvärt sätt i Victor (2013). Han beskriver att när de började att programmera datorer, gjorde de det med maskinkod. Vissa minnesplatser tilldelades värden med hårdkod. Dessa personer såg ner på de som började med Assembler[19]. Det är ingen nytta att slå ihop det datorn ska göra till små kommandon. När utvecklingen sedan gick mot Fortran[20] var det samma visa igen. Assembler folket såg ner på de som höll på med Fortran. Detta visar att utvecklingen går framåt och en del tycker att nya sätt att jobba inte tillför något. De ser inte potentialen i det nya sättet att jobba. Idag arbetar ofta programmerare med moduler som de slår ihop till program. Samtidigt måste de förstå vad de gör för att det ska bli bra. Det gör att använda ett minustecken i en Google-sökning som filtrerar sökningen, kan ses som programmering av moduler. Även fast själva programmeringen inte är så omfattande och de måste ändå veta vad de gör för att det sa bli bra. Problemet är att eftersom Google använder AI för att lista ut vad du är ute efter, blir inte sökningarna absoluta då sökmotorn tolkar in annan data. Det är svårt att se klara samband med det eleven söker på och resultatet.

[18] https://www.gapminder.org/
[19] lågnivå programmeringsspråk, https://en.wikipedia.org/wiki/Assembly_language
[20] Programmeringsspråk https://en.wikipedia.org/wiki/Fortran

Redan Papert (1984) beskrev att programmering ska gå att ta på (tangible) men även andra har gjort det (Grover & Pea, 2013; Sentance & Csizmadia, 2015). Det gjorde han med sina robotar som kunde programmeras med LOGO. Övningarna i denna uppsats är olika tangible. På ena sidan har vi textprogrammering med Silent teacher, på andra sidan har vi blockprogrammering mot hjärnas delar. De är konkretare av två anledningar dels blockprogrammeringen (eleven drar block på skärmen) och dels att eleverna taktilt får arbeta med programmering (arbetar med ett fysiskt objekt, det händer inte bara på skärmen). Smetana och Bell (2012) beskriver hur simuleringar ska vara för att ge ett lärande:

> Computer simulations are most effective when they (a) are used as supplements; (b) incorporate high-quality support structures; (c) encourage student reflection; and (d) promote cognitive dissonance. Smetana och Bell (2012, sid. 1337)

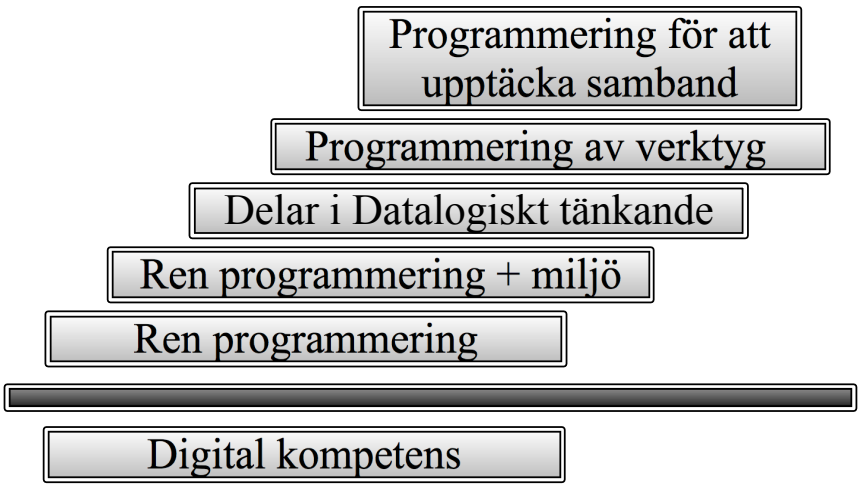
I punkt (b) beskriver de att simuleringarna ska vara både strukturerade ifråga om väl organiserade/vägledda och taktila. Här kan det troligt dras paralleller mot programmering. Att få arbeta med konkreta ting är om inte annat mer motiverande.

Om det inte går att ta på är det bra om det går att visualisera. Sedan programmering började läras ut till barn har mottot vara “low floor, high ceiling” (Grover & Pea, 2013, sid. 40). Att både programmeringen är visuell och resultatet som programmet presenterar går att ta på visar forskning är viktigt (García-Peñalvo, Reimann, Tuul, Rees, & Jormanainen, 2016; Lye & Koh, 2014).

Kapitlet om Transfer (4.5) visar att det är svårt med överföring av kunskap från programmering till andra områden. De studierna har studerat dels mentala förmågor och dels matematik. Det måste finnas en närhet för att det ska kunna ske. Övningen Digitala val har potential för där finns denna närhet. Ett alternativ är att programmeringen används parallellt med det området som ska läras ut. Övningen ”Hjärnas delar” är ett sådant exempel. Annars får man luta sig tillbaka på Pea och Kurland (1984) och konstatera att man blir bra på det man tränar. Håller du på med programmering lär du dig programmering.

6.3. Modell av programmerings relevans för andra ämnen

I kapitel 3 Teori har en sammanfattning gjorts om hur programmeringsundervisningen kan designas utifrån forskning. Tillsammans med diskussionen ovan är tanken med detta avsnitt att presentera och motivera en idé till en modell för att välja ut programmeringsövningar. Modellen är i fem steg som till viss del bygger på varandra.



Figur 3 Beskrivning av modell av relevans för andra ämnen.

För att eleverna ska kunna arbeta med programmering på en dator behöver de en viss digital kompetens (figur 3). I detta fall är det praktiska handhavandet av digitala verktyg som åsyftas, digital kompetens har beskrivits av Skolverket (2016b). Det finns säkert andra kunskaper och förmågor som är nödvändiga och det har till viss del presenterats tidigare i denna uppsats genom Pea och Kurland (1984). De tar där upp Matematisk skicklighet, Analogisk resonerande, Processkapacitet, Villkorsresonerade, Procedural förmåga och Temporalresonerade. Dessa kunskaper och förmågor är fundamentala för att överhuvudtaget komma igång med programmering på en dator. Givetvis är det också beroende på vad man ska göra. Övningar som tränar digital kompetens ligger utanför uppsatsens ram.

Den första nivån är rena programmeringsövningar. Bland övningarna presenterades Silent teacher, ett annat exempel är Hour of code som nämnts tidigare. Eleverna arbetar konkret med programmeringen och utvecklar förhoppningsvis sina programmeringskunskaper. Resultatet av deras kod blir att det händer något på skärmen, en siffra dyker upp, en artefakt rör sig. I dessa övningar är det generellt svårt att se om det kan ske någon transfer till andra ämnen. Som det beskrivits i kapitel 0 om

Transfer behövs det en närakoppling mellan vad de kan och vad de behöver kunna för att ge någon effekt. En generell transfer effekt till andra områden är svår att påvisa då det är svårt att synliggöra den. Det eleven lär sig är i bästa fall är mer programmering.

I nästa nivå gör de samma sak som på första nivån men i en miljö. Det är dock inte troligt att eleverna lär sig något om miljön. Ett exempel skulle kunna vara en bondgård där uppgiften är att få ett får att gå en bit framåt. Detta är en vanlig övning vid introduktion till programmering på tex. Hour of code. Programmeringen är här skiljt från att lära sig något om bondgården. Med andra ord programmeringen kan lika gärna ske i en annan miljö. Det gör att de inte heller kommer att lära sig något om bondgården med hjälp av programmering. För riktigt unga barn och personer som inte behärskar språket kan det dock finnas språkmässiga fördelar såsom att lära sig vad djuren heter. Men det är dock troligt att övningen kan vara mer motiverande att använda metaforen bondgård än att bara flytta boxar. Hour of code bygger på det receptet (programmering + miljö), de har teman hämtade från Minecraft, Frost och Angry Birds.

På den tredje nivån är det övningar som tränar delar i datalogiskt tänkande. Det kan vara när en del tex. mönster (pattern) lyfts ut och eleverna får arbeta med det. Det är vanligt att dessa övningar inte görs på datorer. De kan göra det genom att läsa olika texter och sedan bestämma vilken genre de tillhör, exempelvis vetenskaplig text, en dikt, eller en skönlitterär text. Den här nivån är svår att koppla mot centrala innehållet och kunskapskraven i läroplanen, även om exemplet ovan är ett sådant. De får istället arbeta med syften och allmänna mål, som problemlösning. Därför har ingen sådan övning tagits med i Övningsbanken.

På den fjärde nivån använder eleverna programmering som ett verktyg för att upptäcka ett område, en miljö. Används Lee et al. (2011) modell Use-Modify-Create, är det vanligt att use-nivån används i skolan. Use-nivån går ut på att man använder färdiga program som tex. Word. Dessa program kan inte eleven själva modifiera. Men genom programmet Word kan de upptäcka hur de kan forma en text. Det erbjuder andra möjligheter än att skriva på papper. På modify- och create-nivån är det lite mer arbete som krävs. De kan vara att de skriver program som hjälper till vid sökningar eller hanterar visualiseringar eller statistik. I bondgårdsexempel skulle de kunna programmerar Micro:bit som känner av antal gånger en ko dricker vatten, går ut eller liknade. Enheterna placeras sedan ut på en riktig bondgård och samlar in datan. De kan genom att sammanställa dessa data för att få en förståelse hur en bondgård fungerar. Programmeringen blir ett verktyg för lärande. På denna nivå behöver de kunna programmera, så det är bra kanske till och med nödvändigt att de har arbetat med nivå 1 och 2 innan. En lärare sa ”Scratch är ganska svårt i sig, det är en tröskel som man måste över för att kunna simulera problem i en miljö.”

Programmering för att upptäcka samband är den femte nivån. Ett system är en uppsättning samverkande komponenter som bildar en komplex helhet. Varje system avgränsas av dess rumsliga och tidsmässiga gränser, interaktion med den omgivande miljön, beskrivs av dess struktur och syfte.[21] Dessa komponenter kan mätas och påverkas. Hur de samverkar kan ofta beskrivas med regler, ibland är dessa samband för komplexa. Ser vi på området med ett datalogiskt perspektiv, innebär det att dessa komponenter kan beskrivas som variabler eller instanser och hur de hänger ihop som logik. Det som skiljer denna nivå från nivå fyra är att det som ska läras visas genom regler och samband. På nivå 4 är fokuset på att visualisera undervisningsmålet. Det finns många system som är representerade i en dator, mer eller mindre framgångsrikt.

[21] https://www.merriam-webster.com/dictionary/system Ordbok

Ett exempel är The Human brain project[22] som försöker simulera hjärnan. Som nämndes ligger simuleringar nära till hands när system ska beskrivas i en dator. Är de anpassade enligt Smetana och Bell (2012) kan de ge en god lärande effekt. Simuleringar är ofta färdiga program i datorn, men problemet här är att då behöver inte eleverna sätta sig in vilka regler som styr systemet. Samtidigt kan programmering bli för komplex för att åstadkomma en vettig simulering. Tidigare i denna uppsats har Bogost (2008) beskrivit att arbeta med regler leder till att en förståelse av samband. Begreppet procedurell retorik har använts för att beskriva när eleverna får en förståelse av processen närmar sig det som ska läras. En simulering som Simcity[23] låter användaren bygga upp en stad. Staden är ett komplext system där det finns många komponenter som ska hanteras. Genom att bygga, riva och prioritera vissa komponenter kan användaren få staden att utvecklas.

När det gäller val av övningar finns det många faktorer en lärare måste ta hänsyn till. Pea och Kurland (1984) beskrev fyra olika nivåer en programmerares kompetens kan beskrivas med. Har de inte så stor kompetens behöver övningarna anpassas efter det. Det finns en många olika sätt att programmera på, allt från att skriva ren kod till att arbeta med moduler. Vad som är enklast är svårt att svara på. En del elever föredrar att arbetar med koden. De kan då se vad de gör vid kodskrivningen, det blir tydligt vad ett kommando representerar. Andra föredrar färdiga moduler och de behöver då bara kalla på dessa. Det går ofta fortare att programmera med moduler, då flera kommandon har satts ihop så mycket kod behöver inte skrivas. Allt detta är givetvis avhängigt syftet. Som en lärare skrev ”Det blir så mycket fokus på verktyget så man glömmer bort vad det var för problem man ska lösa egentligen*”* Visualisering av kod och även av resultatet är ett annat sätt att välja övningar på.

> En lärare sa: Däremot tycker jag att Scratch ger en väldigt bra introduktion till strukturen av program. Du har de här repetera blocken och i dom lägger du in vad som ska repeteras och då blir det lite förskjutet. Så får du kodstrukturen grafiskt och är det ett lite mindre steg att börja skriva koden.

NCWIT (2009) betonade att eleverna ska arbeta två och två, för då får man en dialog kring programmering men också kring ämnet eller det som är själva uppgiften.

6.4. Summering

Det finns potential att använda programmering som ett verktyg för lärande. För att kunna undervisa i programmering behöver lärarna ha kunskap i det, dock behövs inte så mycket kunskap som de tror. En lärare tyckte att man behöver tre terminer på högskola. De kan med mindre insatser använda övningar som till viss del använder programmering. Eftersom programmering inte ska ligga i fokus, utan det är bara ett verktyg. Sentance och Csizmadia (2015) håller med men hävdar att de behöver mer tid kompetensutveckling och struktur för att de ska bli bra. Har inte lärarna kunskaper så blir undervisningen oftast praktiskt inriktad, vilket innebär att den teoretiska grunden saknas (Rolandsson, 2015).

[22] http://www.humanbrainproject.eu/en/science/overview/
[23] http://www2.ea.com/sim-city

> De (eleverna[24]) såg också att logiken är liknade när en text i svenskan byggs upp som vid programmering.

Alla övningar en lärare använder i sin undervisning erbjuder olika möjligheter och utmaningar. Ibland kan det vara svårt att se vilken effekt en övning kan ha. På fokusgruppen sa en lärare att det inte är lätt att få eleverna att föra ett naturligt samtal kring som i vårt fall hjärnas centrallob, men programmeringen gör det faktiskt möjligt. ”Det blir också tydligt om de har förstått processen, om de lyckats åskådliggöra det på rätt sätt.” I läroplanen beskrivs också att digitala verktyg ska användas i undervisningen. Digitala verktyg är skapade av programmering. I vissa av dessa verktyg är det programmering som eleven gör när de använder verktyget.[25]

Uppsatsen visar att när lärarna har lärt sig att programmera och implementerat det i sin undervisning är de mer positiva till programmering. De ser även fler möjligheter att använda programmering på olika sätt. Dels kan de variera själva programmerings undervisningen och dels kan de använda det som ett lär-verktyg i andra ämnen. Dock har de svårt att beskriva konkret dessa sätt självständigt. Men fick de resonera kring konkreta övningar såg de flera möjligheter. Till hjälp för denna process har det presenterats en modell som klassificerar programmeringsövningars möjligheter till transfer till andra ämnen. Uppsatsen ska ses som en förstudie och ett ge en möjlig angreppssätt på frågan.

6.5. Metoddiskussion

I Sverige finns det 130000 lärare[26]. I denna uppsats fanns 74 personer via en enkät och ytterligare några via fokusgrupp och intervju. Det gör att respondenterna i denna uppsats representerar en liten del av populationen. Det finns dock en liknade åldersfördelning i denna undersökning av lärare som i populationen lärare, dock har inga signifikanser beräknats. Det tyder på att trots att urvalet inte var slumpmässigt är det troligt representativt i alla fall ur en åldersaspekt. Därför bör resultaten användas med tillförsikt då de är svåra att generalisera utifrån. Uppsatsen ska mer ses som en pilotstudie över området och därigenom visa en möjlig väg att utforska.

Det är många som borde kunnat ha sett länken till enkätundersökningen. En anledningen varför de klickar på den kan förklaras med information-gap teorin. Loewenstein (1994) har beskrivet anledningen varför personer interagerar med länkar.

> the information-gap theory views curiosity as arising when attention becomes focused on a gap in one’s knowledge. Such information gaps produce the feeling of deprivation labeled curiosity. The curious individual is motivated to obtain the missing information to reduce or eliminate the feeling of deprivation. (Loewenstein, 1994, sid. 87)

Så länken till enkäten i sociala medie-grupperna har på något sätt väckt nyfikenheten enligt denna teori. Troligt är att det är de som har någon form av åsikter om programmering som fyllt i den. Detta gör att urvalet kan ha förskjutits. Åsikterna är

[24] förklaring av de
[25] Ett exempel är IFTTT. Som går ut på att om detta händer så ska detta ske. Får jag ett epostmeddelande, så ska en notifiering ske. https://ifttt.com/
[26] Enligt http://siris.skolverket.se/. Databas med statistik över lärare

antagligen för eller emot vilket leder till en polariserad bild av området. För att hantera detta har betoningen varit på exempel och tankar istället för kvantitativ data.

Ett sätt att se på hur det kan generaliseras från en studie med ett litet urval presenteras av Mayring (2007). När datainsamlingsmetoderna som i denna uppsats (fokusgrupp, enkät och intervju) blir de ett urval ur en population. De sammanställs till ett antagande, teori, som leder till en deduktion över området. Det är det som är en generalisering. Problemet med att generalisera är att även fast vi har gjort en aldrig så noggrann studie kan vi inte säga något om den kontext där den ska testas. Så kan vi ha hur många observationer som helst ur en population, det kan alltid finnas ett undantag. Han tar upp en modell med i tre steg för att visa på möjligheten att generalisera utifrån kvalitativa studier. Det första steget är att titta på kontexten, i vårt fall är övningarna som valts ut representativa för sådana övningar som en lärare skulle kunna hitta. När övningarna valts ut till uppsatsen var det en grundprincip, dock har den inte objektivt testats. Det andra steget är att slutsatsen från studien ska en generell ansats. Den ska vara bredare än bara att gälla själva studien. Denna uppsats har lett fram till en modell som kan anses generell. I det sista steget beskriver han longitudinal ansats. I denna uppsats har tre olika observations metoder valts ut vilket på ett sätt stärker denna möjlighet. Dock som tidigare påpekats är dock urvalet litet.

7. Referenser

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: implications for teacher knowledge. *Journal of Educational Technology & Society, 19*(3), 47-57. doi: .

Balanskat, Anja, & Engelhardt, Katja. (2015). Computing our future Computer programming and coding, Priorities, school curricula and initiatives across Europe: European Schoolnet (EUN Partnership AIBSL), Brussels, Belgium.

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

Ben-Ari, Mordechai. (1998). *Constructivism in computer science education.* Paper presented at the Acm sigcse bulletin.

Benton, Laura, Hoyles, Celia, Kalas, Ivan, & Noss, Richard. (2017). Bridging Primary Programming and Mathematics: Some Findings of Design Research in England. *Digital Experiences in Mathematics Education*, 1-24. doi: 10.1007/s40751-017-0028-x

Black, Paul, & Wiliam, Dylan. (2010). Inside the black box: Raising standards through classroom assessment. *Phi Delta Kappan, 92*(1), 81-90.

Bogost, Ian. (2008). The rhetoric of video games. *The ecology of games: Connecting youth, games, and learning*, 117-140.

Bransford, John D., & Schwartz, Daniel L. (1999). Rethinking Transfer: A Simple Proposal with Multiple Implications. *Review of Research in Education, 24*, 61-100. doi: 10.2307/1167267

Burke, Quinn. (2012). The markings of a new pencil: Introducing programming-as-writing in the middle school classroom. *Journal of Media Literacy Education, 4*(2), 121-135.

Cohen, Louis, Manion, Lawrence, & Morrison, Keith. (2011). *Research methods in education [Elektronisk resurs]* (7. ed.). Abingdon, Oxon ; New York: Routledge.

Dalbey, John, & Linn, Marcia C. (1985). The Demands and Requirements of Computer Programming: A Literature Review. *Journal of Educational Computing Research, 1*(3), 253.

Directory, Oxford English. (2017). Computer. Hämtad, från http://www.oed.com.proxy.lnu.se/view/Entry/37975?redirectedFrom=computer - eid

Engelsen, Knut Steinar. (2006). *Gjennom fokustrengsel [Elektronisk resurs] : lærerutdanningen i møte med IKT og nye vurderingsformer*. Bergen: Det samfunnsvitenskapelige fakultet, Institutt for informasjons- og medievitenskap, Universitetet i Bergen.

Flannery, Louise P, & Bers, Marina Umaschi. (2013). Let's dance the "robot hokey-pokey!" Children's programming approaches and achievement throughout early cognitive development. *Journal of research on technology in education, 46*(1), 81-101.

Friberger, Marie Gustafsson. (2015). Gapminder: ett steg mot att använda öppna data i undervisning. *Öppna data i skolan.*Hämtad 3 Maj 2017, från http://oppnadataiskolan.se/2015/11/gapminder-ett-steg-mot-oppna-data-i-undervisning/

García-Peñalvo, F. J. , Reimann, D. , Tuul, M. , Rees, A. , & Jormanainen, I. . (2016). *TACCLE 3, O5: An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers*. Belgium: TACCLE3 Consortium. Hämtad från DOI:10.5281/zenodo.165123.

Greeno, James G, Collins, Allan M, & Resnick, Lauren B. (1996). Cognition and learning. I D. C. Berliner & R. C. Calfee (Eds.), *Handbook of educational psychology* (Vol. 77, s. 15-46). New York, NY, : Macmillan Library Reference Usa; London, England: Prentice Hall International.

Grover, Shuchi, & Pea, Roy. (2013). Computational Thinking in K–12. *Educational Researcher, 42*(1), 38-43. doi: 10.3102/0013189X12463051

Hellmark Knutsson, Helene (2015). *Uppdrag att föreslå nationella it-strategier för skolväsendet*. (U2015/04666/S). Utbildningsdepartementet: Hämtad från http://www.skolverket.se/polopoly_fs/1.240545!/U2015-04666-S_Nationella_it-strategier.pdf.

Ioannidou, Andri, Bennett, Vicki, Repenning, Alexander, Koh, Kyu Han, & Basawapatna, Ashok. (2011). *Computational Thinking Patterns*. Paper presented at the Annual Meeting of the American Educational Research Association, Online Submission, New Orleans, LA. http://files.eric.ed.gov/fulltext/ED520742.pdf

ISTE. (2011). Computational Thinking: A Digital Age Skill for Everyone. Hämtad 6 april, 2017, från https://www.iste.org/explore/articleDetail?articleid=152&category=Solutions&article=Computational-thinking-for-all

Jones, Elizabeth. (2011). *The Trouble with Computational Thinking*. Opublicerat verk. Engl. 890J, Professors Buell and Cooley University of South Carolina. Hämtad från https://www.semanticscholar.org/paper/The-Trouble-with-Computational-Thinking-Jones/1111b4db549ce77290137287069796a228b75ac1

Klahr, David, & Carver, Sharon McCoy. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology, 20*(3), 362-404.

Krueger, Richard A., & Casey, Mary Anne. (2015). *Focus groups : a practical guide for applied research* (5. [updated] ed.). Thousand Oaks, Calif.: Sage Publications.

Lee, Irene, Martin, Fred, Denner, Jill, Coulter, Bob, Allan, Walter, Erickson, Jeri, . . . Werner, Linda. (2011). Computational thinking for youth in practice. *Acm Inroads, 2*(1), 32-37.

Ling, Lo Mun, & Marton, Ference. (2011). Towards a science of the art of teaching: Using variation theory as a guiding principle of pedagogical design. *International Journal for Lesson and Learning Studies, 1*(1), 7-22. doi: doi:10.1108/20468251211179678

Loewenstein, George. (1994). The psychology of curiosity: A review and reinterpretation. *Psychological bulletin, 116*(1), 75.

Lye, Sze Yee, & Koh, Joyce Hwee Ling. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. doi: http://doi.org/10.1016/j.chb.2014.09.012

Mannila, Linda, Dagiene, Valentina, Demo, Barbara, Grgurina, Natasa, Mirolo, Claudio, Rolandsson, Lennart, & Settle, Amber. (2014). Computational Thinking in K-9 Education. *Annual Joint Conference Integrating Technology into Computer Science Education*, 1.

Marton, Ference. (2006). Sameness and Difference in Transfer. *The Journal of the Learning Sciences, 15*(4), 36.

Mayer, Richard E. (2004). Should there be a three-strikes rule against pure discovery learning? *American psychologist, 59*(1), 14.

Mayring, Philipp. (2007). *On generalization in qualitatively oriented research.* Paper presented at the Forum Qualitative Sozialforschung/Forum: Qualitative Social Research.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*: National Academies Press.
NCWIT. (2009). Pair Programming-in-a-Box: The Power of Collaborative Learning. Hämtad 5 maj, 2017, från https://www.ncwit.org/resources/pair-programming-box-power-collaborative-learning
Papert, Seymour. (1984). *Tankestormar : alternativ pedagogik med datorns hjälp* (M. Linder Dannewitz, Översättning.): Stockholm : Forum, 1984 ; (Lund : Beta grafiska).
Pea, Roy D. (1983). *Logo Programming and Problem Solving.[Technical Report No. 12.]*. Paper presented at the Chemeleon in Classroom: Developing roles for computers, Montereal, Canada.
Pea, Roy D. (1987). The aims of software criticism: Reply to Professor Papert. *Educational Researcher, 16*(5), 4-8.
Pea, Roy, & Kurland, Midian. (1984). *On the Cognitive Effects of Learning Computer Programming: A Critical Look. Technical Report No. 9*: Bank Street Coll. of Education, New York N. Y. Center for Children Technology,.
Pirolli, Peter, & Recker, Margaret. (1994). Learning Strategies and Transfer in the Domain of Programming, 235.
Pålsson, Stefan. (2017). Perspektiv på programmering – Omvärldsbloggen. Hämtad 2017-01-10, 2017, från http://omvarld.blogg.skolverket.se/2017/01/10/perspektiv-pa-programmering/
Rolandsson, Lennart. (2015). *Programmed or Not: A study about programming teachers' beliefs and intentions in relation to curriculum.* KTH Royal Institute of Technology.
Sentance, Sue, & Csizmadia, Andrew. (2015). *Teachers' perspectives on successful strategies for teaching Computing in school*. Paper presented at the IFIP TCS 2015 – June 2015.
Sentance, Sue, & Csizmadia, Andrew. (2016). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*. doi: 10.1007/s10639-016-9482-0
Shih, Yu-Fen, & Alessi, Stephen M. (1994). Mental Models and Transfer of Learning in Computer Programming. *Journal of Research on Computing in Education, 26*(2), 154-175.
Skolverket. (2011). Kursplan Samhällskunskap grundskola. Hämtad 19 April, 2017, från https://www.skolverket.se/laroplaner-amnen-och-kurser/grundskoleutbildning/grundskola/samhallskunskap
Skolverket. (2016a). Att planera utifrån kursplanen. Hämtad 13 April, 2017, från https://www.skolverket.se/laroplaner-amnen-och-kurser/grundskoleutbildning/grundsarskola/stodmaterial/att-planera-sin-undervisning-i-teknik/att-planera-utifran-kursplanen-1.173603
Skolverket. (2016b). Redovisning av uppdraget om att föreslå nationella itstrategier för skolväsendet – förändringar i läroplaner, kursplaner, ämnesplaner och examensmål. Hämtad 17 Februari, 2017, från http://www.skolverket.se/om-skolverket/publikationer/visa-enskild-publikation?_xurl_=http%3A%2F%2Fwww5.skolverket.se%2Fwtpub%2Fws%2Fskolbok%2Fwpubext%2Ftrycksak%2FBlob%2Fpdf3668.pdf%3Fk%3D3668
Skolverket. (2017). Alla gymnasieelever ska kunna läsa programmering. Hämtad 4 April, 2017, från https://www.skolverket.se/skolutveckling/resurser-for-larande/itiskolan/alla-gymnasieelever-ska-kunna-lasa-programmering-1.259557
Smetana, Lara Kathleen, & Bell, Randy L. (2012). Computer Simulations to Support Science Instruction and Learning: A critical review of the literature. *International Journal of Science Education, 34*(9), 1337-1370. doi: 10.1080/09500693.2011.605182

Säljö, Roger. (2005). *Lärande och kulturella redskap : om lärprocesser och det kollektiva minnet*. Stockholm: Norstedts akademiska förlag.
The Royal Society. (2012). Shut down or restart? - The way forward for computing in UK schools. Hämtad 6 April, 2017, från https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf
Vaidyanathan, Sheena. (2015, Dec 2, 2015). Computer Science Goes Beyond Coding. Hämtad 25 Apr, 2017, från https://www.edsurge.com/news/2015-12-02-computer-science-goes-beyond-coding
Vee, Annette. (2013). Understanding Computer Programming as Literacy. *Literacy in Composition Studies, 1*(2), 42-64.
Vetenskapsrådet. (2002). *Forskningsetiska principer inom humanistisk-samhällsvetenskaplig forskning* Hämtad från http://www.codex.vr.se/texts/HSFR.pdf
Victor, Bret. (2012). Learnable programming - Designing a programming system for understanding programs. Hämtad 2 Maj, 2017, från http://worrydream.com/LearnableProgramming/
Victor, Bret. (2013). The Future of Programming. Hämtad 12 Maj, 2017, från https://www.youtube.com/watch?v=8pTEmbeENF4
Wibeck, Victoria. (2010). *Fokusgrupper : om fokuserade gruppintervjuer som undersökningsmetod* (2., uppdaterade och utök. uppl. ed.). Lund: Studentlitteratur.
Wing, Jeannette M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.
Yadav, Aman, Mayfield, Chris, Zhou, Ninger, Hambrusch, Susanne, & Korb, John T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), 5.
Åkerfeldt, Anna. (2014). *Didaktisk design med digitala resurser : en studie av kunskapsrepresentationer i en digitaliserad skola*. Stockholm: Institutionen för pedagogik och didaktik, Stockholms universitet.

8. Bilagor

Bilaga 1

"Hej
Jag skriver min magisterexamen på programmet Lärande, kommunikation och IT vid Göteborgsuniversitet. Min tanke är att studera hur programmering kan användas för att ge ett lärande utanför själva programmeringen dvs. hur programmeringen kan bli ett verktyg för lärande. För att göra det kommer jag att bjuda in till en fokusgrupps diskussion. På träffen kommer vi se/pröva några övningar i programmering och diskutera dess relevans. Min förhoppning är att det kan ge dig nya insikter hur programmering kan användas i din undervisning. Hör av dig till mig om du vill komma?

Tidpunkten är 2017-04-27 mellan 16.00 - 17:00 på Linneuniversitet.

mvh

Anders

Bilaga 2

Frågor fokusgrupp

Huvudfrågor var

- Vad lär sig eleven i övningen?
- Hur kan det som lärs ut via programmeringsövningen bidra till att nå kunskapskraven/centrala innehållet i ditt ämne?
- Vilka möjligheter och hinder ser ni för att införa och använda programmeringsövningen som en del i er undervisning?

Kompletterande frågor var

- Använder du programmering i ditt ämne?
- Vad är programmering?
- Vad är datalogiskt tänkande (datalogisk tänkande)?
- Kan ni nämna ett problem som eleverna behöver lösa i ert ämne?
- Hur löser de det problemet?
- Kan datorn vara till hjälp att lösa problemet?
- Kan programmering vara ett sätt att närma sig lärande i ditt ämne?
- Används funktionell och objekt programmering?
- Vilket programmeringsspråk används?
- Analogi matematik - vilka viktiga inslag finns i undervisningen?
- Problemlösning- innehåll vs psykologi

Bilaga 3

Brev till enkät till facebookgrupp. Detta brev vände sig till matematiklärare. Därför har ingressen anpassats till dem.

I matematik är det etablerat vad som ska läras ut, dock har det genom åren har det skett viss förändring. Även sättet som matematik har lärts ut har förändrats.

Det är tänkt att programmering ska införas som en del av läroplanen både för grund- och gymnasieskolan.. Frågan är hur ska programmeringsundervisningen se ut? Vad ska egentligen läras ut? Om 20 år kommer den se likadan ut då?

Jag funderar över dessa frågor i ett arbete och skulle vilja ha din hjälp att svara på några frågor angående detta. Denna enkät undersöker hur lärare ser på att införa och använda programmering i sin undervisning. Det spelar ingen roll om du har använt programmering eller ej, båda perspektiven är viktiga. Enkäten samlar in data till ett magisterarbete. Svaren kommer att redovisas på ett sådant sätt att deltagarna förblir anonyma.

https://goo.gl/forms/xxxxxxxxxxx

Bilaga 4
Bakgrundsenkät till fokusgrupp

Notering
Enkäten har skickats ut elektroniskt som ett webbformulär. Det innebär att designen i detta dokument, inte stämmer visuellt men innehållet är samma.

Programmering i skolan
Programmering ska införas som en del av läroplanen både för grund- och gymnasieskolan. Denna enkät undersöker hur lärare ser på att införa och använda programmering i sin undervisning. Det spelar ingen roll om du har använt programmering eller ej, båda perspektiven är viktiga. Enkäten samlar in data till ett magisterarbete. Svaren kommer att redovisas så att deltagarna förblir anonyma.
Har ni några frågor kring enkäten kan ni kontakta anders.gerestrand@lnu.se
*Obligatorisk
1) Hur viktigt tycker du det är att införa undervisning i programmering i skolan?
❑ Inte viktig ❑ Lite viktigt ❑ Ganska viktigt ❑ Mycket
2) Vilka är de 3 viktigaste delarna i programmering?
3) Vad får eleverna ut av att undervisas om programmering?
4) Har du använt dig av programmering i ditt arbete det senaste året? *
❑ Inget ❑ Lite ❑ Ganska mycket ❑ Mycket

Detta avsnitt kommer respondenten till om de svarat alternativ 1 på fråga 4
Börja programmera
1:5) Vilka utmaningar ser du som lärare att införa programmering som en del i din undervisning?
1:6) Vilka möjligheter ser du som lärare att införa programmering som en del i din undervisning?
1:7) Vad tycker du fokus ska ligga på när programmering lärs ut?
Teori ❑ 1❑ 2 ❑ 3 ❑ 4 ❑ 5 Praktik
1:8) Vilka bra tekniker/strategier/övningar tror du kan hjälpa eleverna att förstå programmering?
1:9) Hur kan det som lärs ut via programmeringsundervisning bidra till att nå andra läroplansmål än de som tar upp programmering?

Detta avsnitt kommer respondenten till om de svarat alternativ 2,3 eller 4 på fråga 4
Bedriva programmeringsundervisning
2:5) Vilka utmaningar ser du som lärare att använda programmering som en del i din undervisning?
2:6) Vilka möjligheter ser du som lärare att använda programmering som en del i din undervisning?
2:7) Vad tycker du fokus ska ligga på när programmering lärs ut?
Teori ❑ 1❑ 2 ❑ 3 ❑ 4 ❑ 5 Praktik
2:8) Vilka bra tekniker/strategier/övningar har du hittat som hjälper eleverna att förstå programmering?
2:9) Hur kan det som lärs ut via programmeringsundervisning bidra till att nå andra läroplansmål än de som tar upp programmering?

Kunskapsnivå
10) Var har du lärt dig om programmering? *
- ❑ Jag kan inte programmera
- ❑ Lärarutbildningen
- ❑ Av-media, media central eller annan privat/kommunal/landstings-bedriven organisation
- ❑ Coder-Dojo
- ❑ Lärt mig själv (Autodidakt)
- ❑ Datavetenskapliga kurser på universitet
- ❑ Andra kurser på universitet
- ❑ Övrigt:

11) Vilken typ av programmeringsutbildning behöver du mer av?
12) Om du skall bedöma dig själv hur skulle du klassificera dig?
- ❑ Programanvändare, har testat lite grann, men använder mest färdiga program
- ❑ Kodskapare, Kan syntaxen och skapa enkla program, läsa och förstå andra program men inga försök till optimering
- ❑ Programskapare, Gör lite mer avancerade program, kan koppla ihop olika delar. Kanske inte helt användarvänliga. Till viss del optimerade
- ❑ Systemutvecklare, kan skriva komplexa och avancerade program. Kan användas av andra personer.

13) Känner du dig trygg att undervisa om programmering?

Inte 1 2 3 4 5 Mycket

14)Din ålder *

❑ 20-30 ❑ 31-40 ❑ 41-50 ❑ 51-

15) Vilka/et ämne/ämnen undervisar du i ? *

- ❑ Matematik
- ❑ Teknik
- ❑ Fysik
- ❑ Kemi
- ❑ Biologi
- ❑ Samhällskunskap
- ❑ Religonskunskap
- ❑ Geografi
- ❑ Historia
- ❑ Svenska
- ❑ Moderna språk
- ❑ Engelska
- ❑ Musik
- ❑ Slöjd
- ❑ Hemkunskap
- ❑ Idrott och hälsa
- ❑ Yrkesämnen på gymnasiet
- ❑ Övrigt:

16) Vilket stadie undervisar du på? *

- ❑ Förskola
- ❑ Lågstadie
- ❑ Mellanstadie
- ❑ Högstadie
- ❑ Gymnasie
- ❑ Vuxenutbildning
- ❑ Övrigt:

17) Övriga kommentarer kring enkäten och programmering

Bilaga 5

Övningsbank- Beskrivning av övningar

1. Operera en hjärna för att lära sig delarna.
   I denna uppgift får eleven skapa en hjärna av papper sedan går uppgiften ut på att koppla in en MakeyMakey för att den ska detektera om eleven kommer åt kanterna när de ska placera plåster på olika delarna av hjärnan. Om eleven kommer åt kanterna kommer en Scratch applikation att markera det. Tanken är att genom att arbeta med hjärnans delar kommer de indirekt lära sig dessa. Programmering blir ett sätt att nå målet.
   - behöver: ledningar, makeymakey, scratch, folie,
   - http://joshburker.blogspot.se/2013/01/makey-makey-scratch-operation-game.html
   - https://scratch.mit.edu/projects/3137739/
   -
2. Chattrobot – gör en egen variant
   Språk är uppbyggt av regler (grammatik). Genom att skapa frågor och svar med hjälp av programmering kan eleven få en förståelse hur språket fungerar. Programmering är blockprogrammering med dra och släpp. Övningen kan göras mer avancerad med hjälp av verktyg som https://twinery.org/ eller http://arisgames.org/
   https://scratch.mit.edu/projects/113323354/

3. Silent teacher (textprogrammering)
   Ett trail and error verktyg för att träna kunskaperna i språket javaskript. Eleven ser en syntax och ska sedan lista ut vad koden ger för svar. Eleven skriver in sitt svar och sedan jämförs. De får grönt om de klarat det annars rött. Vid misslyckande får de en likande fråga.
   http://silentteacher.toxicode.fr/hourofcode

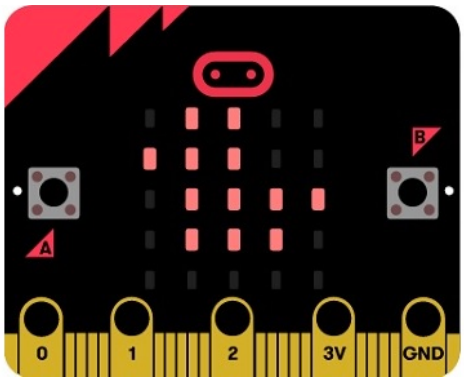
4. Digitala val (till riksdag)
   Är ett sätt att förstå hur komplext det är med digitala val. Tanken är att eleven genom bygga ett valsystem får en insikt i fördelar och nackdelar med att göra val digitalt, men även vad som ska beaktas. Programmeringen görs med Pseudokod och/eller MindMap. Saker som eleverna ska tänka på är
   - beskriv ett röstningsprogram
   - anonym
   - bara rösta en gång
   - säkert
   - enkelt
   - tillgängligt
   - Valhemlighet (ingen annan ska veta vad du röstar på), enskild person ska inte kunna påverkas

5. Google sökning
   De flesta har sökt i Google men inte alla har använt sig av söktekniker som gör att precisionen blir bättre. Övningen visade
   Minus-att välja bort saker som inte ska vara med
   Define: för att hitta definitioner

Filetype: hitta filer av en viss typ tex.pdf eller ppt
Bildsökning: att använda en bild att söka med
Valutaomvandlare: se vad kronan står i dag i förhållande till dollarn.
https://www.google.com/intl/sv_se/insidesearch/tipstricks/all.html

6. Gapminder.
   Är ett verktyg för att få syn på samband. Eleven väljer variabler och upptäcker samband. Det är även möjligt att filtrera data. Programmering består i att tilldela variabler och göra val.
   https://www.gapminder.org/tools/#_locale_id=en;&state_time_value=2000;&marker_select@_geo=kwt&trailStartTime=1850;&_geo=are&trailStartTime=1854;&_geo=swe&trailStartTime=2000;;&axis/_x_domainMin:null&domainMax:null&zoomedMin:null&zoomedMax:null&scaleType=linear;&axis/_y_which=sulfur/_emissions/_per/_person/_kg&domainMin:null&domainMax:null&zoomedMin:null&zoomedMax:null&scaleType=genericLog;;;&chart-type=bubbles

7. Kompass
   I denna övning får eleverna skapa en kompass med hjälp av programmering. Plattformen är Micro:bit som är ett chip med några olika sensorer som kan programmeras. Bland annat en magnetometer som kan visa riktning. Tanken här är att eleverna får en förståelse för hur kompass, grader och elektroniska variant av dessa fungerar.
   https://www.microbit.co.uk/td/lessons/compass



8. Human Resource Machine
   Human Resource Machine är ett iPad spel, där spelaren ska lösa olika arbetsuppgifter. Uppgifterna löser spelaren med programmering. Det utspelar sig i en något dystopisk värld. Det är en chef som är väldigt auktoritär. Och arbetet som ska utföras är monotont. I de centrala målen i för samhällskunskap finns det inskrivet att man ska studera med arbetslivets förutsättningar. Programmering är blockprogrammering med dra och släpp.
   http://tomorrowcorporation.com/humanresourcemachine