GÖTEBORGS UNIVERSITET

Att bygga ett instrument

En instrumentell ansats till programmering som ett verktyg för problemlösning i läromedel för matematik.

Mika Forss
Ämneslärarprogrammet med inriktning mot arbete i gymnasieskolan

Examensarbete: 15 hp
Kurs: LGMA2A
Nivå: Avancerad nivå
Termin/år: VT 2019
Handledare: Johan Wästlund
Examinator: Johanna Pejlare
Kod: VT19-3001-002-LGMA2A

Keywords: Mathematics, Programming, Problem solving, Instrumental approach, Creative reasoning, Textbooks, Sweden

To build an instrument

An instrumental approach to programming as a tool for problem solving in mathematics textbooks

Abstract

Recent changes in the curriculum for the Swedish upper secondary school resulted in an inclusion of programming in some of the mathematics courses. Where included, programming is described as a strategy for mathematical problem solving. Since this is a new area of mathematics in school, how programming is implemented, and its relation to problem solving is an area of interest. This study examines the implementation of programming in textbooks for the first course in mathematics for the science and technology programs. Four different textbooks are included in the study. The research question focuses on how and to what degree the textbooks support students to pick up programming as a tool for problem solving.

To examine this process a framework given by the instrumental approach and instrumental orchestration is used. This includes both the didactical configuration and the exploitation mode in the textbooks. The didactical configuration is analysed using a series of descriptive questions and a classification of tasks. The tasks are classified as mathematical/technical and problem solving or not. In order to determine which tasks to classify as problem solving Lithner's (2008) framework of creative and imitative reasoning is used. The exploitation modes are analysed using a theoretical thematic analysis, focusing on support for the instrumental genesis of the students.

The results show significant differences between textbooks. The absence of problem solving in two of the textbooks is noteworthy. The two textbooks that included problem solving do this in different ways, but the analysis shows that they both lack in the treatment of the technical aspects of programming. Therefore the conclusion is that teachers, regardless the choice of textbook, must act independently in relation to the material in order to fulfil the curriculum and give students an opportunity to claim programming as a tool of their own.

Förord

Jag vill inleda med att rikta ett tack till några som på olika sätt varit del i detta arbete och gjort det möjligt. Ett första tack till min handledare Johan Wästlund för kommentarer och korrekturläsning av arbetet. Tack till Jennie Helge som opponerat på arbetet, och tack också till kursledare Johanna Pejlare som varit examinator och bidragit med tips på litteratur i arbetets tidiga skede. Mina medstudenter Gustav och Oskar ska också ha ett stort tack för att ha agerat bollplank under arbetets gång.

Slutligen ett tack till min familj, Sara och Malte, för att ni stöttat mig och sett till att jag haft annat att tänka på under den här tiden också. Effektiviteten mår bra av lite distraktioner.

Mika Forss

Innehållsförteckning

Figurer och tabeller

1 Inledning

Programmering och matematik har tydliga historiska kopplingar. Med tiden har också programmeringen breddats och blivit en allt viktigare del i vår vardag för att bygga upp och styra den teknik vi använder. Dessa kopplingar lyfts ibland också fram som argument för en integrering av programmering i skolans matematikundervisning. Programmeringen beskriv då som ett sätt att konkretisera matematiken, inspirera elever och ge en djupare förståelse för vissa delar av matematiken.

Genom förändringar i styrdokument blir programmeringen nu också en del av matematikundervisningen i Sverige. Detta sker som en del i en större revidering av styrdokumenten för att förtydliga skolans uppdrag gällande elevernas digitala kompetens. I den digitala kompetensen ingår ett kritiskt förhållningsätt till information och olika typer av media samt användning av digital teknik i olika former. Som ett led i detta skrivs programmering in som en del i matematiken och tekniken för grundskolan. För gymnasiet skrivs programmeringen in som en strategi för problemlösning i vissa kurser.

Min egen erfarenhet från VFU-perioder och annan kontakt med aktiva och blivande lärare är att många är osäkra på sin egen förmåga och kunskap gällande programmering. Bristerna rör då både den egna användningen av programmering och vad de ska göra med programmeringen i matematiken. Frågor som *vilken typ av uppgifter är relevant att arbeta med?* och *hur kan detta förmedlas vidare till elever?* är högst relevanta att ställa sig, framförallt när tidigare erfarenhet eller utbildning saknas.

En möjlig källa för svar på dessa frågor är de läromedel som används. Den stora roll som läromedel har i att forma matematikundervisningen är väl känd. Stor tid ägnas åt arbete med läromedlets uppgifter, och läromedlet fungerar också som den huvudsakliga källan vid planering av undervisningen (Mullis, Martin & Foy, 2008). Det är därför också troligt att läromedel kommer spela en stor roll i hur programmering nu blir en del i matematiken och forma hur lärare utformar sin undervisning.

1.1 Syfte och frågeställningar

Syfte för studien är att undersöka hur programmering inkorporerats i aktuella läroböcker för matematik på gymnasiet. Då området är så outforskat blir rena praktiska aspekter intressanta att lyfta men fokus ligger på implementering i relation till styrdokumentens beskrivning av programmering som en strategi för problemlösning, och därmed hur programmeringen blir ett verktyg som eleverna själva hanterar.

Utifrån detta syfte formuleras följande frågeställning:
Hur och till vilken grad ges eleverna stöd och möjlighet att tillägna sig programmering som ett eget verktyg att hantera för problemlösning i matematik genom läromedel för gymnasiet?

På grund av tidsåtgång för analys och en spridning i vilka kursmaterial som ännu uppdaterats i enlighet med de nya styrdokumenten avgränsas undersökningen till kursen Matematik 1c. Gällande läromedel avgränsas studien också till klassiska läroböcker eller liknande tryckta material avsedda för kusen som helhet. Helt digitala läromedel där också programmering kan utföras integrerat i läromedlet inkluderas därför inte i denna studie. Inte heller material som endast behandlar programmering frikopplat från gymnasiekurserna i matematiks övriga innehåll. Extramaterial som släpps som ett komplement till befintliga kursböcker inkluderas dock.

2 Bakgrund

Nedan kommer bakgrunden till studien att presenteras för att ge ett sammanhang och en teoretisk grund till den undersökning som utförts. Här beskrivs programmeringens roll i svensk skola och tidigare resultat kring programmering och matematik i undervisningen. Sedan kommer ett teoretiskt perspektiv till digitala hjälpmedel att presenteras. En beskrivning av matematisk problemlösning och ramverk för resonemangstyper presenteras här också. Avslutningsvis redogörs kort för läromedels roll i undervisningen.

2.1 Programmering

I likhet med den förändring som skett i digital teknik, och användningen av den, har också betydelsen av ordet programmering skiftat med tiden. Från att ha syftat på all datoranvändning föras betydelsen i och med utvecklingen av mer intuitiva applikationer som blev allt vanligare (Blackwell, 2002). Detta ledde till att personer utan expertis gällande just datorhantering också kunde använda datorer, och programmering skiljs därmed från datoranvändande i allmänhet. Blackwell visar vidare hur programmering i tidig praktik var nära kopplat till det matematiska området, för att senare lägga allt med fokus på grundläggande operationer, symbolspråk och lösningsmetoder. Programmering gick på detta sätt från att vara matematiskt till att röra mer generell databehandling. Utvecklingen av programmering för olika användningsområden breddar denna beskrivning ytterligare, och Blackwood argumenterar för att en tydlig gräns för vad som är programmering och inte därmed är svår att dra. En mer vardaglig styrning av teknik kan därför också inkluderas i begreppet.

Vad som åsyftas med programmering är på detta sätt inte självklart, men då denna studie vill fånga den användning av programmering som nu införts i skolan används programmering här för att syfta på skapandet av instruktioner, i flera steg som används för att styra en dators agerande. De miljöer som används ska också syfta till skapandet av dessa instruktioner. Användning av kalkylblad och andra program där kommandon kan användas på ett programmeringsliknande sätt, som symbolhanterande räknare och GeoGebra, inkluderas inte i denna beskrivning

2.1.1 Programmering i skolan historiskt

På liknande sätt har programmeringens roll i svensk skola förändrats. Starten på denna process kan hittas under 1970-talet då programmering börjar introduceras i gymnasieskolan. Tidigare, under 1960-talet, har programmering bara varit en del av yrkesutbildning. Rolandsson och Skogh (2014) beskriver detta som en första fas där programmering tas in i skolan. Genom försöksverksamheter börjar grunden för en läroplan i datakunskap utvecklas och Rolandsson och Skogh beskriver hur programmeringens roll då var omdebatterad. Frågan gällde då om programmering är något för en bredare grupp av elever, eller bara för vissa med fallenhet för det. Fokus låg på programmering som en del av andra ämnen, exempelvis matematik, och hur det kan användas i större sammanhang för att angripa problem. Införandet motiverades också med att programmeringens praktiska aspekter skulle kunna locka fler elever att läsa matematik och naturvetenskapliga ämnen.

En andra fas beskrivs sedan i och med införandet av en läroplan för datakunskap under 1980-talet (Rolandsson & Skogh, 2014). Programmeringen, och dess mer tekniska aspekter som syntax och kommandon, hade då en framträdande roll i undervisningen. Med tiden breddas elevgrupperna som ska läsa datakunskap, först genom testverksamheter på samhällsvetenskaplig linje, och sedan genom stöd för inköp av datorer för göra det möjligt att

erbjuda datalära och programmering till flera. Rolandsson och Skogh beskriver också hur innehållet i samband med detta förändras, och programmeringens roll blev allt mindre framträdande. Istället lades fokus på logiskt tänkande och datorers möjligheter och begränsningar. I och med den nya läroplanen 1993 plockades programmering bort från den allmänna datakunskapen, och undervisades istället endas i separata kurser av yrkeskaraktär.

2.1.2 Programmering i skolan idag

Programmering är sedan höstterminen 2018 inskrivet som en del av ämnena matematik och teknik på grundskolan, och finns förutom kurser inriktade mot programmering och angränsande ämnen också inskrivet som en del i vissa matematikkurser på gymnasiet. Beskrivningen nedan kommer röra de kopplingar till programmering som återfinns i matematik från grundskola till gymnasium.

2.1.2.1 Grundskola

I grundskolan finns programmering inskrivet som en del av matematikämnets övergripande syfte, och som innehåll för årskurserna 1-3, 4-6 och 7-9. I ämnets syfte beskrivs hur eleverna ska

> ges möjligheter att utveckla kunskaper i att använda digitala verktyg och programmering för att kunna undersöka problemställningar och matematiska begrepp, göra beräkningar och för att presentera och tolka data.
>
> (Skolverket, 2018)

I det centrala innehållet under punkten algebra återfinns sedan följande punkter för respektive årskurser:

> Åk 1-3
> Hur entydiga stegvisa instruktioner kan konstrueras, beskrivas och följas som grund för programmering. Symbolers användning vid stegvisa instruktioner.
>
> Åk 4-6
> Hur algoritmer kan skapas och användas vid programmering. Programmering i visuella programmeringsmiljöer.
>
> Åk 7-9
> Hur algoritmer kan skapas och användas vid programmering. Programmering i olika programmeringsmiljöer.
>
> (Skolverket, 2018)

För årskurserna 7-9 inkluderas också programmering under problemlösning med punkten

> Hur algoritmer kan skapas, testas och förbättras vid programmering för matematisk problemlösning.
>
> (Skolverket, 2018)

I kommentarmaterialet till matematik (Skolverket, 2017c) lyfts progressionen mellan årskurserna fram. För åk 1-3 läggs fokus på grundläggande förståelse för programmering utifrån konkreta situationer, för att eleverna i senare årskurser ska kunna använda programmering allt mer som ett verktyg för matematiken. Progression gällande programmeringsmiljöer lyfts också

fram. Från att enbart gälla visuella programmeringsmiljöer i åk 4-6 kan också textbaserade programmeringsmiljöer introduceras i åk 7-9.

2.1.2.2 Gymnasieskolan

I gymnasieskolans matematikkurser återfinns programmering som en del av problemlösning inom kurserna Matematik 1c, 2c, 3b, 3c, 4, 5 och Specialisering (Skolverket, 2017a). Programmering tas alltså upp på de kurser som läses på naturvetenskapligt och tekniskt program, samt de högre kurserna på övriga högskoleförberedande program. Programmering tas endast upp i följande punkt, som är identisk för alla kurser där programmering ingår.

> Strategier för matematisk problemlösning inklusive modellering av olika situationer, såväl med som utan digitala verktyg och programmering.
>
> (Skolverket, 2017a)

I Skolverkets kommentarmaterial till ämnet matematik (2017b) beskrivs hur den formulering som används är medvetet öppen. Detta för att möjliggöra en variation i vilken omfattning programmering förekommer, och i vilka former. Däremot poängteras att det är ”ett krav att programmeringen används som en strategi för problemlösning” (Skolverket, 2017b). Det lyfts här också att inga specifika krav ställs på programmeringsspråk eller miljöer. Kalkylblad nämns här också som en möjlig metod för att arbeta på programmeringsliknande sätt med elever som saknar relevanta förkunskaper i programmering. Samtidigt poängteras dock att elever, i möjligaste mån, ska få använda relevanta programmeringsmiljöer, vilket inte inkluderar kalkylblad (Skolverket, 2017b).

Några exempel på hur programmering kan användas som verktyg för problemlösning lyft också. Dessa är simulering för uppskattning av sannolikhet, gissningar som en systematisk strategi och för att ”utforska problem av typen ’för vilka heltal mellan 500 och 1000 gäller att…’” (Skolverket, 2017b).

2.1.3 Programmering och matematik

Kopplingen mellan matematik och programmering är tydlig ur ett historiskt perspektiv, och detta har under åren också framhållits i utbildningssammanhang. Programmeringen har då beskrivits som en möjlighet att konkretisera matematiken och utveckla abstrakt tänkande. Ett exempel återfinns i Feurzeig, Papert & Lawler (2011), ursprungligen presenterats 1968, där programmeringsspråket LOGO presenteras som är gjort för att användas som en del i skolmatematiken. Några av möjligheterna med programmering som lyfts fram här är matematiska experiment, ett ramverk för matematiskt tänkande, övning i problemlösning och insikter i vissa matematiska begrepp, som variabler och funktioner. I och med återinförandet av programmering som en del i matematiken är det intressant att titta på två mer aktuella studier av relationen mellan programmering och matematik.

Med anledning av programmeringens införande som en del i algebra för grundskolan analyserar Kilhamn och Bråting (*kommande*) relationen mellan *algebraiskt tänkande* och *datalogiskt tänkande* (eng: *computational thinking*). De identifierar några viktiga skillnader i användningen av symboler och beskriver hur matematik och programmering kan ha olika betydelser för samma symbol, samma betydelse men olika symboler eller så saknas motsvarigheter mellan de två områdena. Betydelsen av vissa begrepp lyfts också fram som en viktig skillnad, exempelvis variabel och funktion. Kilhamn och Bråting lyfter fram att dessa skillnader kan orsaka problem, speciellt för elever som redan har svårigheter inom dessa områden. De framhåller dock också

att skillnaden mellan de två områdena också kan skapa tillfällen för lärande. Användande av flera olika symboler i olika sammanhang kan, om det hanteras medvetet av lärare, ge elever en möjlighet att se förbi en specifik symbol och skapa förståelse för de bakomliggande matematiska innebörderna. På samma sätt kan också programmeringen bidra med en breddning i förståelsen av ett visst begrepp och dess innebörd, som kanske inte kommer fram lika tydligt annars. Nyckeln här är då att skillnader mellan användningen i programmering och matematik ”noteras, konstrasternas och diskuteras” (Kilhamn & Bråting, *kommande,* s. 7, egen översättning).

Misfeldt och Ejsing-Dunn (2015) använder istället en instrumentell ansats, beskrivs under nästa rubrik, för att studera aktiviteter där programmering introduceras till elever i yngre åldrar. Fokus ligger på i vilken mån programmeringen bidrar till att lyfta fram matematiska begrepp och tankar hos eleverna. De tillskriver här lärarens agerande som ledare av aktiviteten en stor betydelse. Genom att lyfta fram de matematiska idéerna och använda matematiska koncept och principer för att hjälpa eleverna framåt i sin förståelse gör läraren det möjligt för eleverna att faktiskt ta del av programmeringens positiva aspekter. Hur läraren hanterar programmeringen är dock avgörande för att detta ska ske (Misfeldt & Ejsing-Dunn, 2015).

2.2 Instrumentell ansats

Nedan kommer en instrumentell ansats att presenteras. Detta är ett teoretiskt perspektiv som kan användas för att belysa användningen av verktyg, och som ofta tillämpas på just digitala hjälpmedel i matematiken. Trouche och Drijvers (2014) beskriver hur den instrumentella ansatsen (eng: *the instrumental approach*) har sitt ursprung i fransk kontext, och där används för studier av lärande i teknikrika miljöer som matematikundervisning med symbolhanterande räknare. De poängterar att ordet instrumentell här inte ska förstås som “användande av regler utan orsak” (Trouche & Drijvers, 2014, s. 4) utan istället handlar om ett meningsskapande av ett verktyg.

2.2.1 Artefakt och instrument

Utgångspunkten för den instrumentella ansatsen är en syn på kunskapsbildning som något som inte bara kan ske i relation till mentala strukturer, utan också sker i relation till praktiska objekt och funktionella redskap. Detta beskrivs av Verillon och Rabardel (1995) som vidare visar hur dessa *artefakter* kan agera medlare i relationen mellan ett subjekt och objektet för subjektets handlingar. Här poängteras också hur artefakten inte kan användas som den är, utan subjektet behöver tillägna sig artefakten och integrera den i sitt arbete, och på så sätt göra artefakten till ett *instrument* i subjektets hand. Drijvers et al. (2010) menar att vi kan tala om ett instrument först när en meningsfull relation existerar mellan artefakten och subjektet i relation till objektet för dennes handlingar.

Ett exempel skulle kunna vara en hammare som ska användas för att slå ned en spik. Hammaren är då den artefakt som subjektet, personen som håller i hammaren, vill använda för att uppnå ett objekt, i detta fall att slå ner spiken. Subjektet behöver då också någon kunskap om vad en hammare är och hur den kan användas. Exempel på detta kan vara vad den kan användas till, vilken ände som man ska hålla i, vilken sida av hammarhuvudet som ska träffa spiken, och så vidare. Först då bildas ett instrument som kan användas för att uppnå objektet, och personen kan använda hammaren för att slå ner sin spik.

Vad som är artefakten i en given situation behöver dock inte vara självklart, speciell när mer komplexa digitala verktyg används. Drijvers et al (2010) lyfter hur dynamiska geometriprogram, exempelvis GeoGebra, kan ses som en artefakt i sin helhet, men också som en samling av artefakter, en för att konstruera något, en för att dra i något och så vidare. Då ett instrument byggs av en artefakt i relation till ett objekt, ett syfte, så kommer också en komplex artefakt att ge upphov till en mängd olika instrument (Trouche, 2005). Ett exempel på detta kan vara en grafisk miniräknare, som kan ge upphov till ett instrument för att studera grafen till en funktion, ett annat för numerisk lösning av ekvationer, och så vidare.

2.2.2 Instrumentell genesis

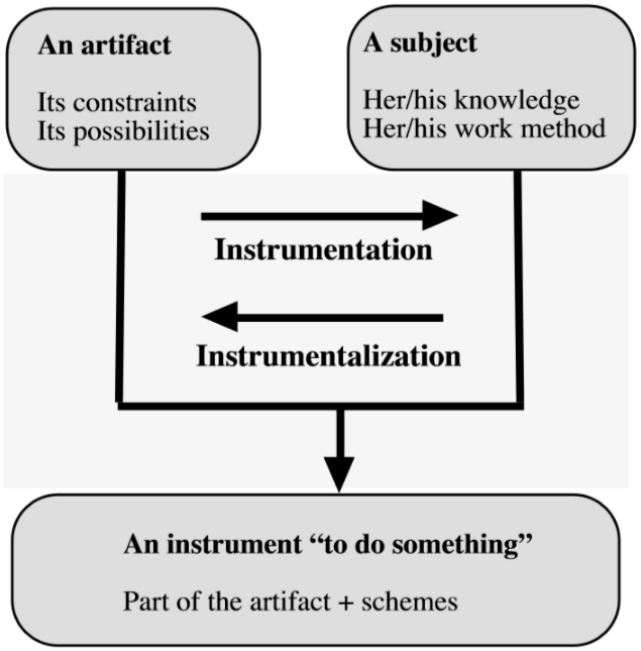
Den process där ett instrument bildas utifrån en artefakt benämns *instrumentell genesis* (Verillon & Rabardel, 1995). Vikten av den instrumentella genesisen visas på av Guin och Trouche (1998) som studerat användningen av digitala hjälpmedel i matematiken. De presenterar ett exempel på hur tillgången till digitala hjälpmedel inte nödvändigtvis är ett stöd för elever. Grupper av elever har då fått i uppgift att beräkna följande gränsvärde:

$$\lim_{x \to \infty} \ln x + 10 \sin x$$

Av de elever som inte haft tillgång till digitala hjälpmedel svarade alla korrekt, att uttrycket går mot oändligheten. Av elever som haft en grafritande räknare till hjälp var det istället endast 10% av eleverna som klarade uppgiften. Detta kan förklaras då den graf som visas när uttrycket ritas ut i ett standardfönster på en grafritande räknare visar en kraftig variation och inte indikerar någon tydlig trend. Eftersom eleverna inte vet hur de ska hantera sitt digitala hjälpmedel på ett givande sätt i situationen blir alltså hjälpmedlet istället ett hinder. Utifrån denna och vidare studier av hur elever tillägnar sig digitala hjälpmedel beskriver Guin & Trouche den instrumentella genesisen därför som en både nödvändig och komplex process, som kan innehålla många komponenter samt se olika ut för olika elever. Detta poängteras också av Drijvers el al (2010) som beskriver den instrumentala genisisen som en ”ständigt pågående, icketrivial och tidskrävande utveckling” (s. 108, egen översättning).

Rabardel och Bourmaud (2003) menar att ett instrument består både av en artefakt och ett schema för användning, två skilda men sammankopplade komponenter. Utifrån detta menar de också att den instrumentella genesisen kan förstås som två parallella processer: *instrumentering* och *instrumentalisation*, där instrumentering är en utveckling och förändring av scheman för användning och instrumentalisation är en utveckling av artefakten.

Beskrivningen av den instrumentella genesisen som två processer vidareutvecklas sedan av Trouche (2005) som lyfter fram artefaktens och subjektets ömsesidiga påverkan på varandra. Instrumentering beskrivs därmed som den process där artefakten påverkar subjektet, och instrumentalisation som den process där subjektet påverkar artefakten. Genom dessa processer kommer också ett instrument bildas i syfte att utföra en viss typ av uppgift. Detta instrument kommer bestå av delar av artefakten samt de scheman som utvecklats hos subjektet. Se figur 1 nedan.



Figur 1: En schematisk bild över den instrumentella genesisen och de två komponenterna instrumentering och instrumentalisation.

(Trouche, 2005, s. 144)

Instrumentering sker genom att artefakten med sina begränsningar och möjligheter formar subjektet, vilket resulterar i scheman för hur subjektet hanterar artefakten. Dessa scheman kan sedan delas upp i två kategorier, *användningsscheman* och *instrumentella handlingsscheman.* Användningsscheman är då riktade mot artefakten, medan instrumentella handlingscheman är riktade mot objektet för aktiviteten. (Trouche, 2005). Då scheman framförallt är mentala blir de svåra att observera, och utifrån detta lyfter Trouche vidare fram *gester* och *tekniker* som de delar av användningsscheman respektive instrumentella handlingsscheman som blir synliga när artefakten ska användas.

Ett exempel på detta skulle kunna vara en grafritande miniräknare som ska användas för att lösa en ekvation. Att starta räknaren, ställa in ritfönster och liknande är då gester som används, handlingar som är riktade mot artefakten. Att skriva in ekvationens högra och vänstra led som funktioner för att sedan använda verktyget *intersect* är istället en teknik som används, då handlingen här är riktad direkt mot objektet, att lösa ekvationen.

Trouche (2005) beskriver vidare hur instrumentalisation kan beskrivas i fyra steg. *Upptäckande, selektion, personalisering* och *transformering.* Upptäckande och selektion sker då av artefaktens olika möjliga funktioner. Personalisering och transformering handlar sedan om att artefakten formas på olika sätt, där personalisering handlar om mindre anpassningar av artefakten utifrån subjektets preferenser. Transformering är större förändringar av artefakten, som skapande av nya funktioner eller en annan användning av artefakten än det den ursprungligen är utformad för. Som helhet blir instrumentalisationen på detta sätt ett "uttryck för subjektets specifika aktivitet: vad en användare tänker att verktyget är designat för och hur det borde användas" (Trouche, 2005, s. 148, egen översättning).

2.2.3 Instrumentell orkestrering

Den instrumentella genesisen behöver alltså ske hos varje individ som vill tillägna sig ett verktyg, men denna process sker också i ett sammanhang. För elever i skolan sker detta i ett klassrum med en lärare som skapar förutsättningar för denna process att ske. För att kunna beskriva de förutsättningar och det stöd som ges för denna kollektiva genesis kan begreppet *instrumentell orkestrering* användas (Drijvers et al. 2010). Instrumentell orkestrering syftar till extern styrning av den instrumentella genesisen överlag, men tillämpas ursprungligen på en lärares styrning av elever i ett klassrum.

Instrumentell orkestrering introduceras som begrepp av Trouche (2004) som därigenom vill visa på den nödvändiga externa styrningen av elevers instrumentella genesis, exempelvis av läraren i en klassrumssituation. Trouche menar också att den instrumentella orkestreringen kan beskrivas genom *didaktiska konfigurationer*, som val av artefakter och design av omgivning, och *användningsformer* (eng: *exploitation modes)*, hur dessa konfigurationer utnyttjas i syfte att främja elevernas instrumentella genesis.

Drijvers (2010) beskriver de didaktiska konfigurationerna som de grundläggande förutsättningarna för inlärningen, som tillgängliga verktyg, placering i klassrum, material att bearbeta med mera. Användningsformerna blir då hur läraren agerar för att utnyttja de resurser som finns tillgängliga. Hur interagerar läraren med elever och tillgängliga artefakter, och hur eleverna ska använda de artefakter som finns tillgängliga. Drijvers (2010) lägger också till ytterligare en aspekt av den instrumentella orkestreringen, *didaktiskt utförande*. Därmed skiljer Drijvers den mer övergripande planeringen av ett lektionsinnehåll från det specifika och spontana som uppstår när en lektion genomförs. Exempel på detta skulle kunna vara oväntade frågor eller problem som uppstår under en lektion eller andra spontana ageranden från läraren.

2.3 Problemlösning och resonemang

Ett viktigt första steg i att närma sig problemlösning är att definiera vad som menas med ett problem, då betydelsen kan skifta från grundläggande matematiska uppgifter till utmanande forskningsfrågor (Schoenfeld, 2016). I denna studie används Schoenfelds (1985) definition, där denna beskrivning återfinns.

> Being a ‘problem’ is not a property inherent in a mathematical task. Rather, it is a particular relationship between the individual and the task that makes the task a problem for that person. The word problem is used here in this relative sense, as a task that is difficult for the individual who is trying to solve it. Moreover, that difficulty should be an intellectual impasse rather than a computational one. [...] If one has ready access to a solution schema for a mathematical task, that task is an exercise and not a problem
>
> (Schoenfeld, 1985, s. 74)

Vad som är ett problem avgörs alltså en uppgifts relation till den som ska lösa den. För att vara ett problem ska uppgiften innebära en intellektuell utmaning, där färdigt lösningsschema saknas.

En ingång till att avgöra vad som kan klassas som problemlösning blir då att analysera den typ av resonemang som uppgiften kräver av eleverna. Denna ingång används av Brehmer, Ryve och Steenbrugges (2016) i en analys av problemlösning i svenska läroböcker för

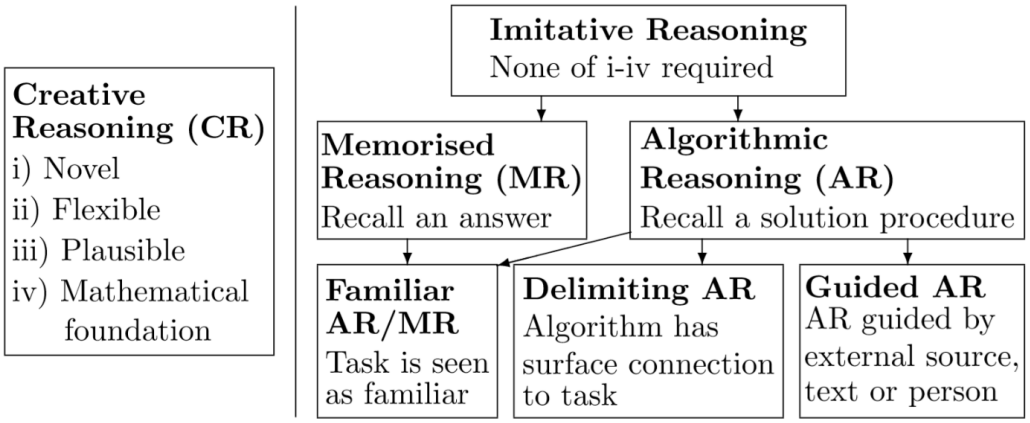
gymnasiematematiken. Här likställs problemlösning med uppgifter som kräver *kreativt resonemang* utifrån Lithners (2008) ramverk för olika typer av resonemang i matematik.

2.3.1 Resonemang

Lithner (2008) tar sin utgångspunkt i de olika resonemang elever använder när de löser uppgifter. Resonemang syftar då på det sätt att tänka som används för att motivera påståenden och nå slutsatser. Här syftas alltså på resonemang i en bredare mening, inte något som måste vara grundat i logik eller resultera i giltiga slutsatserna. Huvudsaken är att eleven som för resonemanget agerar efter det. På detta sätt inkluderas all det tänkande som elever behöver göra i normala matematikuppgifter (Lithner, 2006).

Utifrån analys av elevers lösningar av uppgifter identifierar Lithner två huvudgrupper av resonemang, *kreativt resonemang* och *imitativt resonemang* (2006, 2008). Kreativt resonemang kännetecknas då av ett innehåll som sedan tidigare är okänt, där resonemang måste föras grundat i matematiken. Med imitativt resonemang är det istället möjligt att lösa uppgiften utan detta genom att återskapa ett svar eller en lösningsalgoritm.

Nedan, i figur 2, visas skiljelinjen mellan dessa två typer av resonemang. Här är också beskrivningen utbyggt med några punkter som karaktäriserar det kreativa resonemanget, och en uppdelning av olika typer av imitativt resonemang. Dessa kommer beskrivas vidare nedan.



Figur 2: Lithners ramverk för matematiskt resonemang, med distinktion mellan kreativt och imitativt resonemang.

(Lithner, 2006, s. 5)

2.3.1.1 Kreativt resonemang

Lithner (2008) beskriver tre krav för kreativt resonemang. Den första av dessa är att resonemanget är något nytt och en, för eleven, ny tankegång bildas eller återskapas från att varit bortglömd. Relaterat till uppgifter innebär det alltså att uppgiften måste vara ny för eleven, och inte heller direkt kopplad till tidigare erfarenheter. Den andra punkten är rimlighet (eng: *plausbility*) i resonemanget. I detta hänvisar Lithner (2008) till Pólyas värdering av resonemang. Ett rimligt resonemang handlar då om att skilja mellan olika gissningar och värdera dem. Det behöver alltså inte vara ett formellt, logiskt hållbart resonemang, men det ska ändå vara grundat, och därmed rimligt. Den tredje punkten är en matematisk grund för resonemanget, alltså att resonemanget ska grundas på matematiska egenskaper för de komponenter som ingår.

Denna beskrivning av kreativt resonemang ligger nära Schoenfelds (1985) beskrivning av ett problem. De fokuserar båda på vad en uppgift kräver av en elev. Detta beskriver också Lithner (2008) men poängterar också att kreativt resonemang, till skillnad från problemlösning, inte behöver vara en utmaning.

2.3.1.2 Imitativt resonemang

Imitativt resonemang skiljer sig sedan från kreativt resonemang genom att de inte kräver de tre punkterna som beskrivits ovan. Lithner (2008) beskriver sedan hur detta kan delas in två huvudtyper av imitativt resonemang, *memorerat resonemang* och *algoritmiskt resonemang*. Memorerat resonemang består då i att komma ihåg svaret direkt. Algoritmiskt resonemang består istället av att en lösningsalgoritm kopplas till uppgiften, som sedan kan användas för att nå fram till ett svar. Kopplingen mellan algoritm och uppgift kan dock ske på några olika sätt, vilket ger upphov till tre typer av algoritmiskt resonemang (AR).

En första sådan är *bekant AR* (eng: familiar AR). Här känns uppgiftstypen igen för att sedan kopplas till en algoritm som ger en lösning. I *begränsande AR* väljs istället en algoritm utifrån uppgiftens ytliga innehåll, utan att tanke om varför denna algoritm skulle ge det sökta svaret. I en uppgift där en funktion är given kan eleven på detta sätt testa att derivera eller integrera, oavsett problemformulering. Slutligen beskrivs också *guidad AR* där ytliga likheter mellan uppgiften och ett exempel eller en formel hittas, och den givna lösningsalgoritmen följs för att nå en lösning.

2.4 Läromedel

Intresset för att studera framställningen av programmering i läromedel grundar sig i den framträdande roll dessa har i svensk skola. TIMMS 2007 (Mullis, Martin & Foy, 2008) visar att läromedel används som den primära grunden för undervisningen för 95 % av elever i åk 8 i svensk skola. Vidare menar Pepin och Haggarty (2001) att läromedel kan agera som faktisk läroplan i skolan, alltså det som faktiskt når eleverna, och Johansson (2003) lyfter hur läromedel i många avseenden definierar för skolelever vad matematik är för någonting.

Johansson (2005) identifierar också några punkter genom vilka läromedel påverkar undervisningen. Inledningsvis presenteras det matematiska innehållet i läromedel mest troligt av lärare, samtidigt som innehåll som inte inkluderas mest troligt inte presenteras. Vidare influeras lärare ofta av läromedlets upplägg, och läromedlet används som primär källa när lärare bestämmer sig för hur innehåll ska presenteras. Läromedel är alltså en viktig del i den undervisning som faktiskt når eleverna, och blir därför intressant att studera.

3 Metod

Den metod som använts i studien kommer här att presenteras. Inledningsvis beskrivs urval och genomförande, medan ett större fokus sedan läggs på den analys som genomförts. Under analys beskrivs också hur de teoretiska ramverken som presenteras i bakgrunden kommer användas för att analysera materialet. Slutligen lyfts också frågor om giltighet och forskningsetiska principer.

3.1 Urval

Som grund för urval av läromedel att analysera har de förlag som publicerat de tidigare dominerande läromedlen används. Statistik över läromedel finns tyvärr ej tillgängligt i Sverige, men serierna från förlagen Natur & Kultur (Matematik 5000), Liber (M-serien), Gleerups (Exponent) och Sanoma Utbildning (Origo) tycks allmänt accepterade som dominerande. Utifrån information från respektive förlags hemsida har sedan material valts ut för analys. Från förlagen Natur & Kultur och Liber hittades nya uppdaterade serier, Matematik 5000+ respektive Numerus, som uppgavs vara anpassade till förändringarna i kursplanerna gällande programmering. Från förlagen Gleerups och Sanoma Utbildning hade inga nya läromedel som inkluderar programmering släppts, utan istället hade kompletterande material publicerats gällande detta, vilka kommer användas som material för analys. Då studien begränsats till material rörande kursen Matematik 1c kommer de läroböcker som är avsedda för den kursen att ingå som datamaterial, samt de delar av extramaterialen som är riktade mot samma kurs. De material som ingår i studien är då:

Numerus 1c	Rung, A., Von Heijne, E., & Rundlöf, T. (2018)
Matematik 5000+ 1c	Alfredsson, L., Heikne, H., Holmström, B., Dyrander, J., & Karlsson, M. (2018)
Extramaterial till Exponent	Gleerups(*u.å.*a) – Exempel Gleerups(*u.å.*b) – Övningar
Extramaterial till Origo	Sanoma(*u.å.)*

Vidare kommer läromedlen att hänvisas till genom namn på läromedelsserien. För Numerus och Matematik 5000+ kommer sidhänvisning att inkluderas när specifika uppgifter eller citat lyfts fram. Då sidnumrering saknas för Exponent och Origo används här endast namn eller numrering på aktuellt exempel eller uppgift. Länkar till extramaterial återfinns i referenslistan och är öppna att ladda ner. Från Natur & Kulturs hemsida är det också möjligt att ladda ner de delar av Matematik 5000+ som berör programmering.

3.2 Genomförande

Studien genomfördes sedan genom att de delar av materialen som berör programmering identifieras och plockas ut. Vidare samlades också alla uppgifter in som datamaterial. Analysen genomfördes utifrån såväl materialet som helhet som enskilda uppgifter. Uppgifter klassificerades med hjälp av ramverk som redovisas nedan under analys. Materialet som helhet analyserades med hjälp av analysfrågor för att fånga praktiska aspekter. En mer ingående kvalitativ analys genomfördes sedan för att lyfta fram det stöd som läromedlen ger eleverna i

att tillägna sig programmering som ett verktyg för problemlösning. Denna analys genomfördes som en teoretisk tematisk analys, vilket också beskrivs nedan under analys.

3.3 Analys

För att kunna söka svar på frågeställningen om stöd och möjlighet för elever att tillägna sig programmering som ett eget verktyg kommer en instrumentellt ansats att användas som grund för analysen. Programmeringsspråket och dess utvecklingsmiljö som helhet ses då som artefakt. Subjektet är eleven och objektet för elevens handling ses som matematisk problemlösning. Detta gör det möjligt att analysera förutsättningar för instrumentell genesis av programmering just som verktyg för problemlösning.

Utifrån den instrumentella ansatsen är det då aktuellt att studera den instrumentell orkestrering som kommer till uttryck i läromedlen. Av den instrumentella orkestreringens tre nivåer, didaktiska konfigurationer, användningsformer och didaktisk utförande är didaktiska konfigurationer och användningsformer möjliga att använda för analys av läromedel. Vid användning av läromedel i ett klassrum är dock det didaktiska utförandet en viktig del, men en analys av detta går utanför denna studies ramar.

3.3.1 Didaktiska konfigurationer

De didaktiska konfigurationerna tar sig för läromedel uttryck i läromedlets allmänna struktur, valet av innehåll, typ av uppgifter och liknande praktiska aspekter. För att analysera detta kommer följande frågor att användas:

1. *Vilket programmeringsspråk används?*
2. *Vilka matematiska områden berörs?*
3. *Hur många sidor ägnas åt programmering?*
4. *Behandlas programmeringen separat eller integrerat?*
   Separat - Helt separat material eller ett eget avsnitt
   Semi-integrerat – Integrerat men i tydligt avgränsade sektioner
   Integrerat - Integrerat med det övriga innehållet
5. *Hur stor del av uppgifterna är av matematisk respektive teknisk karaktär?*
6. *Hur stor del av uppgifterna är problemlösning?*

För fråga 1-4 är det möjligt att söka svar direkt i materialet, men för fråga 5-6 behövs ramverk för hur klassificering sker. Detta kommer behandlas nedan. För att fånga ytterligare aspekter gällande didaktiska konfigurationer som kan ha missats i ovanstående analysfrågor kommer också en jämförelse göras mellan läroböckerna för att synliggöra andra eventuella likheter och skillnader.

3.3.1.1 Klassificering utifrån karaktär

Genom en förstudie observerades en skillnad mellan uppgifterna, där vissa saknar egentligt matematiskt innehåll eller svårighet för eleverna och uppgiften är på så sätt av en rent teknisk karaktär. I stort sett alla uppgifter som ingår i denna studie innehåller tekniska aspekter, men syftet med denna analysfråga är att lyfta fram de uppgifter som enbart är tekniska. Därför definieras här uppgifter av *teknisk karaktär* som uppgifter helt utan matematiskt innehåll eller problemsituation som behöver förstås, eller där det matematiska innehållet bör vara självklart för eleverna sedan tidigare. Övriga uppgifter klassas då som av *matematisk karaktär*. Två exempel på denna klassifikation följer nedan.

> Skriv ett program som frågar efter antalet fåglar och den procentuella minskningen. Om antalet fåglar är 747 stycken och minskningen är 5 % varje år ska programmet skriva ut följande resultat:
> Hur många fåglar finns det i kolonin?
> 747
> Med hur många procent minskar antalet varje år?
> 5
> Antalet har halverats efter 14 år.
>
> (Matematik 5000+, Upp. 5, s. 83)

Denna uppgift klassificeras som teknisk då det nya för uppgiften ligger i att värden ska kunna matas in till programmet. Resten av lösningen är givet i ett exempel på samma uppslag. Det matematiska innehållet hos uppgiften kan alltså betraktas som självklart.

> Skriv ett program som beräknar sannolikheten att få en kåk då man drar fem kort helt slumpvis från en kortlek med 52 kort. En kåk innebär att man har fått tre kort av en valör och två av en annan valör.
>
> (Exponent, Upp. 109)

Denna uppgift klassificeras som matematisk då den kräver en förståelse av problemsituationen för att kunna angripas. Uppgiftens lösning kan alltså inte ske genom rent tekniska lösningar, utan en analys av situationen krävs.

3.3.1.2 Klassificering utifrån resonemang

För att avgöra vad som kan klassas som problemlösning krävs en teoretisk ram. I likhet med Brehmer, Ryve & Steenbrugges (2016) läroboksanalys gällande mängden problemlösning i svenska läroböcker i gymnasieskolans matematik kommer Lithners (2008) ramverk för kreativ och imitativt resonemang att användas. Utifrån Brehmer, Ryve & Steenbrugge (2016) och Lithner (2008) klassas då uppgifter som kräver kreativt resonemang som problemlösnings uppgifter. Uppgifter som går att lösa med imitativt resonemang klassas därmed inte som problemlösning. Utifrån beskrivningen av imitativt resonemang kan detta identifieras genom att uppgiften går att lösa med hjälp av närhet till exempel och andra uppgifter, att en färdig algoritm finns, eller med hjälp av gissningar utifrån uppgiftens ytliga innehåll, utan att resonemang om rimlighet för lösningsmetod behövs.

I likhet med Brehmer, Ryve & Steenbrugges (2016) räcker det här också med att en del av uppgiften kräver kreativt resonemang för att uppgiften ska klassas som problemlösning. Då kreativt resonemang enligt Lithners beskrivning innehåller en matematisk grund för resonemanget kommer uppgifter som klassas som problemlösning vara en delmängd av de som klassas som matematik till sin karaktär. Det är alltså matematisk problemlösning som syftas på när en uppgift klassas som problemlösning. Nedan följer några exempel på klassificering ur materialet för att exemplifiera hur ramverket används.

> Du ska välja en ny kod till ditt kodlås. Skriv ett program som slumpar fram 4 siffror mellan 0 och 9 och skriver ut dem på skärmen.
>
> (Numerus, Upp. 3, s. 158)

Denna uppgift klassas inte som problemlösning då det på samma uppslag finns beskrivning av hur ett tal slumpas och detta utvecklats också till att slumpa fram två tal i föregående uppgifter.

Förändring av intervall för slumpvalet har också berörts, vilket innebär att uppgiften inte innehåller några nya aspekter utan går att lösa direkt med det eleven har framför sig. Uppgiften går alltså att lösa med imitativt resonemang.

> Låt programmet i uppgift 1 lösa följande ekvationer.
> a) $3x - 5 = 26$ b) $5(4x + 1) = 45$ c) $16 - 3x = 4x - 5$
>
> (Matematik 5000+, Upp. 2, s. 155)

Inför denna uppgift är ett program som löser ekvationer av typen ax+b=c givet till eleverna. Uppgiften klassas inte som problemlösning det framgår i uppgiften samt beskrivningen av programmet alla steg som behöver göras. Uppgiften kan därför lösas med imitativt resonemang.

> Ett tal där summan av talets alla delare (utom talet självt) är lika med talet självt, kallas för ett perfekt tal. Talet 6 är delbart med 1, 2, 3 och 6. Summan av delarna (utom 6) är 1 + 2 + 3 = 6. Talet 6 är alltså ett perfekt tal. Skriv ett program för att testa om ett tal är perfekt.
>
> (Origo, Perfekta tal)

Denna uppgift klassas som problemlösning då ingen tidigare beskrivning av tillvägagångsätt finns tillgängligt för eleven, och eleven själv kommer behöva resonera med en matematiskt grund gällande delbarhet och hur delare till ett tal kan identifieras. Kreativt resonemang behövs för att lösa uppgiften.

3.3.2 Användningsformer

I läromedlen tar sig användningsformer uttryck i hur de didaktiska konfigurationerna används för att stödja elevernas instrumentella genesis. Detta kan exempelvis vara det stöd som ges till eleverna i arbetet med uppgifter och hur uppgifterna används, hur artefakten och dess olika aspekter presenteras samt de scheman och tekniker som eleverna förväntas utveckla och använda i arbetet.

För att analysera de användningsformer som kommer till uttryck i läromedlen har en kvalitativ tematisk analys av materialet genomförts. Utgångspunkter för analysen är det teoretiska ramverk som ges av den instrumentella ansatsen. Analysen kan därför genomföras som en teoretisk tematisk analys utifrån Braun och Clarke (2006). En tematisk analys beskrivs här i sex steg. (1) Bekanta sig med materialet, (2) Skapa initiala koder, (3) Leta efter teman, (4) Pröva teman, (5) Definiera teman och (6) Rapportskrivning. Analysen utgår från frågeställningen, och den instrumentella ansatsen. Kodning och bildning av teman sker alltså utifrån dessa. Ett tema är då enligt Braun och Clarke något som representerar en del av innehållet i materialet och fångar aspekter som är viktiga i relation till den aktuella frågeställningen.

Fokus i denna analys ligger på de användningsformer som blir synliga i materialet, och förutsättningar för en god instrumentell genesis hos eleven. Andra viktiga begrepp för analysen blir då instrumentering, och instrumentalisation, samt beskrivningarna av dessa. Speciellt då uppdelningen av scheman i användningsscheman och instrumentella handlingsscheman.

3.4 Giltighet

Resultatet från de inledande analysfrågorna ges giltighet genom en närhet till material och användningen av ramverk för klassificering för de mer innehållsrika frågorna. Under klassificeringen har också en andra bedömning inhämtats från annan lärarstudent insatt i ramverket i de enstaka fall där tveksamhet uppstod. Reliabiliteten i bedömning utifrån Lithners

ramverk stöds också av Brehmer, Ryve & Steenbrugges (2016). Validiteten i att klassa uppgifter som kräver kreativt resonemang som problemlösning kan dock ifrågasättas. Lithner (2008) påpekar också att kreativt resonemang inte nödvändigtvis behöver vara en utmaning för eleven på samma sätt som problemlösning. Vissa av uppgifterna som klassats som problemlösning kan alltså kräva kreativt resonemang men inte vara en tillräcklig utmaning för att vara problemlösning. Detta torde dock vara en begränsad del av uppgifterna, och metoden ger därmed fortfarande en bild av förekomsten av problemlösning med programmering i läromedlen som helhet.

Giltigheten i den tematiska analysen ges framförallt av en systematiskt utförd kodningsprocess och sökande efter teman som sedan prövats mot materialet. Resultatet från denna del av analysen gör inte anspråk på att fånga alla aspekter som är möjliga, men de som identifierats har giltighet. Analysen har här också medvetet analyserats utifrån en teoretisk ram för att lyfta fram aktuella aspekter. Andra ingångar till materialet hade på så sätt kunnat lyfta fram något annat.

3.5 Forskningsetiska aspekter

Studien har genomförts utifrån Vetenskapsrådet (2002) riktlinjer för forskningsetiska principer. Då allt insamlat datamaterial i studien är offentligt publicerat, påverkas inte någon enskild person av analysen och de punkterna som Vetenskapsrådet beskriver under individskyddskrav blir på så sätt oproblematiska.

4 Resultat

Resultatet från analysen av materialet kommer här att presenteras. Inledningsvis presenteras resultaten utifrån de begrepp som används som ingång för analysen, didaktiska konfigurationer och användningsformer. Under dessa rubriker lyfts resultat ur materialet som helhet samt jämförelse mellan olika läromedel. För en tydligare överblick och en sammanfattning presenteras därefter också resultatet från analysen utifrån varje enskilt läromedel.

4.1 Didaktiska konfigurationer

De didaktiska konfigurationerna i läromedlen presenteras nedan för två nivåer. Allmänna konfigurationer som rör läromedlet som helhet, och sedan de uppgifter som är del i läromedlet.

Några första allmänna likheter mellan läromedlen är att alla har valt att använda Python som programmeringsspråk, och den omfattning som programmeringen ges i läromedlen i relation till hela läromedlets omfång är också likartad, där 8-13 sidor av materialet berör programmering. Placeringen skiljer sig dock åt. Exponent och Origo behandlar programmeringen i ett helt separat material, medan Matematik 5000+ och Numerus behandlar det semi-integrerat. Detta då programmeringen är uppdelat mellan lärömedlets kapitel, men också alltid är tydligt utmärkt som ett eget avsnitt, vanligtvis i slutet av ett kapitel.

Viss spridning finns också i de matematiska områden som berörs. Talteori och sannolikhetslära är de två områden som behandlas i alla läromedel, därefter finns en större variation. Exempel på vad som tas upp i övrigt är funktionsbegreppet, procentuell förändring, aritmetik och numerisk approximation.

Läromedlen skiljer sig också i tillgången till facit. För Exponent och Matematik 5000+ finns inget facit tillgängligt, till skillnad från Numerus, där facit för alla uppgifter som berör programmering finns tillgängligt för eleverna i slutet av läromedlet. I materialet från Origo finns lösningsexempel i tillhörande lärarhandledning till varje aktivitet, alltså inte direkt tillgängligt för eleverna men möjligt för läraren att dela ut.

Läromedel	Sidor*	Sidor programmering	Programmeringsspråk	Placering
Exponent	390	9	Python	Separat
Matematik 5000+	411	13	Python	Semi-integrerat
Numerus	404	12	Python	Semi-integrerat
Origo	334	8	Python	Separat

Tabell 1: Beskrivning av läromedlens övergripande struktur med avseende på programmering.
***För läromedel med externt material redovisas här sidantal för ursprungligt läromedel.**

En variation mellan läromedlen har också observerats gällande de uppgifter som återfinns i materialet. Uppgifter av teknisk karaktär är mer framträdande i Numerus och Matematik 5000+, medan uppgifter av matematisk karaktär tar en större plats i Origo och Exponent, där Exponent sticker ut med 8 av 10 uppgifter av matematisk karaktär. En liknande spridning återfinns också för uppgifter som klassats som problemlösning. Här är skillnaden också mer framträdande då Matematik 5000+ inte har någon sådan uppgift och Numerus endast har 1 problemlösningsuppgift av 34. Detta kan då kontrasteras mot Origo och Exponent där 10 av 39 respektive 6 av 10 uppgifter är problemlösning.

Läromedel	Uppgifter	Teknisk	Matematisk	Problemlösning
Exponent	10	2	8	6
Matematik 5000+	28	21	7	0
Numerus	34	31	3	1
Origo	39	20	19	10

Tabell 2: Resultat från klassificering av uppgifter i materialet.

Relevant för uppgiftsanalysen av Origo är också att detta läromedel består av aktiviteter som sedan bryts ned i delfrågor riktade till eleverna. Dessa delfrågor är de som har används som uppgifter och klassificerats i analysen. Deluppgifterna ingår därmed i sammanhang som utgörs av den aktuella aktiviteten. Som helhet innehåller alla dessa aktiviteter delar som klassats som av matematisk karaktär och delar som är problemlösning.

4.2 Användningsformer

Resultatet av denna del kommer redovisas utifrån tre aspekter av användningsformer i läromedlen, samt hur dessa aspekter tar sig uttryck i materialet. Den första är uppgifternas roll, den andra förutsättningar för instrumentering och den tredje förutsättningar för instrumentalisation.

4.2.1 Uppgiftstyper

I materialet identifieras tre olika typer av uppgifter som blir synliga i uppgifternas relation till det övriga materialet. Dessa är *bakåtsyftande*, *problemorienterade* och *aktivitetsstödjande* uppgifter. I materialet har det också blivit tydligt att dessa tre typer av uppgifter på olika sätt relaterar till distinktionen mellan matematisk och teknisk samt problemlösningsuppgifter, därför berörs också dessa kopplingar nedan.

Bakåtsyftande uppgifter återkopplar tydligt till de exempel eller den förklarande text som föregått dem. Uppgiften kan på så sätt vara att använda ett kommando som presenterats, testa förståelsen av något begrepp eller att testa använda ett program som blivit givet. Mindre ändringar av tidigare given kod i syfte att öka förståelsen av ett kommando, eller en variabels påverkan är också en förekommande variant. Dessa bakåtsträvande uppgifter kan också kopplas till uppgifter av såväl matematisk som teknisk karaktär, men har en tydlig närhet till uppgifter som går att lösa med imiterande resonemang. De är alltså inte problemlösning. Ett exempel på detta kan vi se i uppgiften nedan. Här syftar uppgiften helt tillbaka till en kod som tidigare är given och en mindre förändring ska här ske för att upprepa for-loopen flera gånger.

```
For n in range(0, 10):
      a = random.randint(1,2)
      if a==1: odd =  odd + 1
      else: even = even + 1
print('Udda:', odd, 'Jämna:', even)
```

Modifiera programmet så att 100 slumptal testas. Notera fördelningen mellan udda och jämna tal.

(Numerus, Upp. 4, s. 159)

Aktivitetsstödjande uppgifter är istället framåtsyftande för att hjälpa eleven vidare i arbetet med en större problemformulering eller förståelsen av en situation. Detta kan innebära mindre delfrågor som steg för steg bygger upp en helhet, specialfall av situationen att pröva eller tips om tekniska lösningar att testa. Denna typ av uppgift har ingen tydlig koppling till uppdelningen mellan matematisk och teknisk eller huruvida uppgiften är problemlösning, utan här finns en stor spridning. Exempel på detta hittar vi i aktiviteten Eratosthenes såll, delar av denna visas nedan. Här får eleven veta målet med aktiviteten och efter att ha fått metoden för Eratosthenes såll presenterad delas uppgiften upp i mindre delar. Dessa delar handlar om att förstå algoritmen som ska användas, men visar sedan eleven vilka de första stegen kan vara. Frågorna visar därmed på en riktning framåt i arbetet med uppgiften.

> I den här aktiviteten får du skriva ett program som skriver ut alla primtal från och med 2 till och med ett tal *n*.
> […]
> 1 Utför algoritmen för hand där *n*=15
> […]
> 2 Skriv ett program som skapar en lista med alla heltal från och med 2 till och med 100.
> 3 a) Skriv ett program som tar bort alla jämna tal större än 2 från listan som du skapade i uppgift 1. Kom ihåg att talet 2 ska vara kvar i listan eftersom det är ett primtal.
> b) Vilket är nästa primtal efter talet 2?
> […]
>
> (Origo, Eratosthenes såll)

Slutligen är problemorienterade uppgifter mer fristående och ger eleverna utrymme att på ett självständigt sätt angripa problemformuleringar som är nya för eleverna. Exempel och förklaringar som finns i anslutning till uppgiften behandlar här inte samma typ av situation och specifika lösningsmetoder. Istället fungerar exempel som en allmän guide i hur problem kan angripas och behandlas, och som allmänna exempel på hur en programkod kan se ut. Den mer självständiga karaktären på dessa uppgifter gör att de har en koppling till uppgifter som kräver kreativt resonemang, och är alltså problemlösning. Ett exempel på detta visas nedan. Uppgiften innehåller flera aspekter som är nya för eleven, då inga exempel eller tidigare uppgifter berört delbarhet och hur det kan arbetas med i Python. Eleven behöver alltså angripa problemformuleringen utan en tydlig ingång, och alla aspekter av att faktiskt använda programmering som ett verktyg för problemlösning inblandas.

> Skriv ett program som avgör om ett inmatat tal är ett primtal.
>
> (Exponent, Upp. 102)

4.2.2 Instrumentering

Instrumentering beskrivs av den instrumentella ansatsen som en process där scheman byggs upp hos subjektet, där ett schema beskriver hur en artefakt kan användas för att uppnå ett visst syfte. Dessa scheman delades sedan in i användningsscheman, som riktar sig mot artefakten, och instrumentella handlingsscheman som riktar sig mot objektet för handlingen. I materialet har stöd för att utveckla dessa scheman visats på två sätt. Den första är implicit, där eleverna genom exempel får se hur dessa scheman uttrycks i gester och tekniker. Den andra är explicit, där scheman uttryckta i gester och tekniker förklaras direkt för eleverna.

4.2.2.1 Användningsscheman

Stöd för utveckling av användningsscheman är synligt i materialet både implicit och explicit och variationen mellan läromedel är stor. Den synliga delen av dessa användningscheman, gester, behandlas implicit genom exempeluppgifter. Här är fokus för uppgiften någonting annat, men i lösningen visas också på olika gester och deras funktion. Hur dessa gester fungerar förklaras dock inte, utan kan bara anas av sammanhanget.

Ett exempel på detta är exempeluppgiften *Ankomst till mötesplats* i Exponent som i lösningsprogrammet använder kodraden

```
Anna = tidAnna * random.random()
```

för att generera ett slumpmässigt reellt tal mellan 0 och variabel tidAnna. Varför denna kod gör detta, och vad `random.random()` innebär förklaras aldrig för eleven, men exemplet presenterar ändå hur ett slumpmässigt reellt tal inom ett intervall kan genereras.

Ett annat exempel på implicit stöd för utveckling av gester återfinns i en exempelkod från Matematik 5000+

```
t = float(input("Ange reaktionstiden (s): "))
# Läser in reaktionstiden
v = float(input("Ange hastigheten (km/h): "))
# Läser in hastigheten
v = v/3.6
# Enhetsomvandlar
s = v * t
# Beräknar reaktionssträckan
print("Reaktionssträcka (m): ", s)
# Skriver ut reaktionssträckan
```

(Matematik 5000+, s. 58)

Här ska programmet i de två första stegen läsa in två reella tal, vilket görs med kommandot `float(input())`, där `input()` läser in ett värde, och `float()` omvandlar detta så att Python kan tolka det inmatade värdet som ett reellt tal. Den allmänna funktionen för kodraden framgår av sammanhanget och kommentarerna inskrivna i koden (märkta med #). Vad kommandot `float()` respektive `input()` har för funktion beskrivs dock inte för eleven mer ingående.

Ett exempel på hur samma kommandon som i exemplet ovan behandlas explicit återfinns i Numerus.

> Om programmet innehåller raden
>
> ```
> x=int(input('Mata in x:'))
> ```
>
> blir du uppmanad att skriva in ett värde för x. Programmet väntar tills du har skrivit in valfritt heltal och tryckt Enter, innan det fortsätter.
>
> `int` betyder att programmet vill att du anger ett heltal. Heltal heter integer på engelska. Vill du kunna mata in tal med decimaler byter du `int` mot `float`,

> som står för *floating point number*, eller *flyttal* i svensk översättning. Ett flyttal kan vara ett decimaltal eller ett tal i grundpotensform. Flyttal är ett approximativt (ungefärligt) sätt för datorn att hantera reella tal.
>
> (Numerus, s. 222)

Här ges eleven i en förklarande text en mer ingående bild av funktionen hos aktuella kommandon. Hur koden förändras för att anpassas till en annan situation tas också upp.

Spridningen mellan läromedel är stor gällande behandlingen av gester. Numerus sticker ut genom explicit behandling, medan Exponent och Matematik 5000+ behandlar detta implicit. Origo avviker också från övriga läromedel genom att knappt behandla gester alls, varken explicit eller implicit.

4.2.2.2 Instrumentella handlingsscheman

Instrumentella handlingsscheman, och de tekniker som är den synliga delen av dessa, skiljer sig från användningsscheman och gester genom att de är riktade mot objektet för handlingen. Det kan därför vara värt att påpeka att denna analys utgår från matematisk problemlösning som objekt, där programmeringsspråket är artefakt. Utifrån detta har tekniker kunnat identifieras i materialet på två olika nivåer. Dessa är generella tekniker för problemlösning med hjälp av programmering och mer specifika tekniker för att möta olika typer av problem.

En explicit behandling av generella tekniker återfinns i Exponent och Matematik 5000+. Detta är allmänna problemlösningsstrategier som eleverna uppmuntras använda, och som också används i lösningen av exempeluppgifter i de båda läromedlen.

I Exponent beskrivs denna strategi så här:

> Vi delar upp lösningen i tre delar, analys, pseudokod och programkod. […]
> Pseudokod är ett sätt att skriva en algoritm, d.v.s. ett antal instruktioner som löser ett problem, utan att behöva fundera över hur språkets syntax ser ut.
>
> (Exponent, Exempel - Kast med sex tärningar)

Den strategi som presenteras i Matematik 5000+ liknar Exponents, men med en lite annan struktur. En avslutande punkt om att testa och värdera programmet är också inkluderad.

> En problemlösningsstrategi för programmering
> 1. Förstå
> 2. Planera
>   A Resultat C Variabler
>   B Lösning D Algoritm
> 3. Genomföra – Koda
> 4. Testa och värdera
>
> (Matematik 5000+, s. 56)

Mer specifika tekniker berörs inte explicit i materialet men utifrån de uppgifter som ingår kunde tre olika specifika tekniker identifieras. Dessa är att *simulera slumphändelser, pröva med villkor* och *iterativt generera*.

Simulering av slumphändelser berör här slumphändelser som är både kontinuerliga och diskreta. Exempel på diskreta situationer kan vara simulering av kast med flera tärningar eller

någon typ av kortspel. Kontinuerliga slumphändelser kan som exempel istället vara ankomst till en busstation, där två personer ankommer inom varsitt tidsintervall, där alla tidpunkter inom detta intervall är lika troligt.

Prövning med villkor handlar oftast om att testa alla möjliga alternativ utifrån ett visst villkor för att identifiera värden eller situationer som uppfyller dessa. Exempel på detta kan vara att hitta alla delare till talet 14 genom att beräkna resten när 14 divideras med alla tal mindre än sig självt, eller att hitta pythagoreiska tripplar genom att undersöka för vilka heltal under 100 som uppfyller ekvationen $a^2 + b^2 = c^2$. En variant kan också vara att urvalet av situationer som ska prövas mot villkoret genereras iterativt. Här kan syftet vara att undersöka hur många steg som behöver tas innan ett visst villkor är uppfyllt. Ett exempel kan vara att undersöka när en population når ett visst värde, givet en procentuell förändring per år.

Iterativt genererande används istället i syfte att närma sig ett sökt värde, utan ett självklart villkor för när önskat värde nåtts. Ett exempel på detta är numeriska approximation av ett irrationellt tal, som en approximation av $\sqrt{2}$ med formeln $x_{n+1} = \frac{x_n + 2/x_n}{2}$.

Numerus skiljer här ut sig från de övriga läromedlen genom att inte beröra någon av dessa tekniker. Såväl Exponent som Matematik 5000+ har innehåll som kopplas till att simulera slumphändelser och pröva med villkor, medan endast Origo inkluderar alla tre. Värt att poängtera är dock att alla dessa tekniker berörs implicit, genom uppgifter och exempel som innehåller dessa aspekter. I Matematik 5000+ behandlas tekniker framförallt i exempel, och inte i uppgifter på samma sätt som i Exponent och Origo.

4.2.3 Instrumentalisering

I materialet har också identifierats hur delar av instrumentaliseringsprocessen kan se ut gällande programmering för problemlösning. Det första steget, upptäckande, är synlig i upptäckande av olika kommandon i kodspråket och hur dessa fungerar. Selektion kommer sedan till uttryck när eleverna kombinerar kommandon och skriver en kod för att uppnå ett visst syfte. Den egna formuleringen av hur ett program ska fungera och dess olika komponenter ses sedan som ett uttryck för en personalisering, där eleven själv behöver lägga grunden för programmets struktur. Det sista steget, transformering, har inte blivit synligt i materialet men skulle kunna ske genom att eleven på egen hand behöver hitta bibliotek att bygga ut programspråkets funktioner med som är lämpliga för den aktuella problemsituationen.

I läromedlen var dessa steg av instrumentalisering synliga både genom stöd till processen och genom utrymme för eleven att ta steg vidare. En variation mellan läromedlen är också synlig. Numerus berör endast upptäckande, Matematik 5000+ upptäckande och selektion och Exponent och Origo innehåller möjlighet för såväl upptäckande och selektion som personalisering.

Ytterligare en aspekt av instrumentaliseringen ges av att inte bara titta på vilka delar av artefakten som ska användas, utan också när artefakten som helhet är rätt verktyg för problemet. Denna aspekt berörs inte av något av läromedlen i studien. Det framgår alltså tydligt i sammanhanget att programmering är det verktyg som eleverna förväntas välja för sin problemlösning. I vissa fall används också programmering för att lösa uppgifter där andra verktyg som finns till elevernas förfogande är mer lämpade. Ett exempel på detta kan vara i Matematik 5000+ där en exempeluppgift går ut på att skriva ett program som löser en ekvation på formen ax+b=c (s. 154). Ser vi endast till målet att lösa ekvationen med ett digitalt

hjälpmedel är en symbolhanterande räknare eller motsvarande ett effektivare och kraftfullare verktyg, som också ingår i kursen Matematik 1c (Skolverket, 2017a).

4.3 Läromedlen

Nedan följer en beskrivning av de fyra läromedlen för sig utifrån de aspekter som berörts ovan. Beskrivningen går inte in på detaljer utan istället fokuseras på de tydligt framträdande karaktärsdragen för att ge en bild av läromedlet som helhet.

4.3.1 Exponent

Här behandlas programmering i ett helt separat material, där materialet består av en samling exempel och uppgifter. Andelen uppgifter av matematisk karaktär och problemlösningsuppgifter är relativt hög, och 6 av 10 uppgifter har klassats som problemlösning. I sitt sammanhang karaktäriseras uppgifterna i Exponent av att vara problemorienterade, de ger alltså eleverna utrymme att på egen hand angripa problem.

Förutsättningarna för instrumentering varierar sedan för de olika nivåerna. Gester berörs endast implicit i exempelkod, förklaringar för olika kommandon eller liknande förekommer alltså inte. Sett till generella tekniker presenterar Exponent en uttalad strategi för problemlösning med programmering som sedan används i exempeluppgifterna. Av de mer specifika teknikerna berör uppgifterna i Exponent simulering av slumphändelser och pröva med villkor.

Gällande instrumentalisering berörs de tre första stegen, upptäckt, selektion och personalisering. Stödet i exempel är dock tydligast för de senare två, selektion och personalisering, genom att problemformuleringar i exempel angrips i flera steg, vilket gör det möjligt för eleven att följa problemlösningen från analys till ett färdigt program.

4.3.2 Matematik 5000+

I Matematik 5000+ är programmeringen semi-integrerad och materialet är fördelat mellan läromedlets olika kapitel men behandlas alltid i tydligt avgränsande avsnitt. Det generella upplägget består av exempel vilka följs av en samling uppgifter. En relativt stor andel av uppgifterna är av teknisk karaktär, 21 av 28, och ingen av uppgifterna är problemlösning. I sitt sammanhang är uppgifterna också bakåtsyftande, de är alltså tätt kopplade till exempel och förklarande text där eleverna endast får testa en kods funktion eller göra mindre förändringar.

Vareriande förutsättningar finns sedan för instrumenteringsprocessen. Gester behandlas endast implicit genom exempelkod som stödjs av förklarande kommentarer. En generell teknik behandlas explicit genom en problemlösningsstrategi för programmering. Av de mer specifika teknikerna för problemlösning berörs simulering av slumphändelser och pröva med villkor, och dessa berörs framförallt i exempel. Stöd för instrumentalisation är mindre synliga och endast de två första stegen, upptäckt och selektion, berörs av materialet.

4.3.3 Numerus

Här behandlas programmering semi-integrerat. Materialet som helhet består av förklarande text med kortare exempel, vilket följs av uppgifter. Uppgifterna är nästan uteslutande av teknisk karaktär, och endast en uppgift är problemlösning. I sitt sammanhang är uppgifterna

bakåtsyftande i sin karaktär, och prövar olika tekniska aspekter som berörts i den förklarande texten.

Förutsättningarna för instrumentering är avgränsade, men utförliga. En explicit behandling av gester präglar hela materialet. Tekniker behandlas inte i läromedlet, varken generella eller mer specifika. På liknande sätt rör förutsättningarna för instrumentalisation endast det första steget av upptäckt. Överlag saknas koppling till objektet, problemlösning nästan helt.

4.3.4 Origo

I Origo behandlas programmering i ett separat material som är strukturerat i aktiviteter. Till varje aktivitet finns sedan ett blad som ska lämnas ut till eleverna samt en lärarhandledning med exempel på lösningar och stöd kring genomförande av aktiviteten. Aktiviteterna som helhet hade alla inslag av problemlösning, och av delfrågor som ingick klassades 19 av 39 som matematisk till sin karaktär. Av dessa är tio stycken problemlösning. I sitt sammanhang agerar dessa delfrågor aktivitetsstödjande, de guidar alltså eleven vidare genom aktiviteten.

Stödet till instrumenteringsprocessen är här tydligt avgränsat. Gester berörs knappt i materialet, varken implicit eller explicit, utan förutsätts hos eleverna. Någon generell teknik behandlas inte och då aktiviteterna redan är nedbrutna i delfrågor begränsas elevens utrymme att själv utveckla en sådan. Av de mer specifika teknikerna för problemlösning berörs alla tre tidigare nämnda, alltså simulering av slumphändelser, pröva med villkor och iterativ generering. Förutsättning för instrumentalisation finns gällande de tre första stegen, upptäckt, selektion och personalisering, med ett fokus på de två senare.

5 Diskussion

Nedan kommer resultaten i studien att diskuteras för att lyfta fram skillnader och likheter mellan läromedlen. Läromedlen kommer också sättas i ett klassrumssammanhang för att lyfta fram didaktiska konsekvenser, och tydliggöra de olika läromedlens styrkor och svagheter vid användning. Resultatet sätts här också i relation till bakgrunden. Vidare kommer studiens metod och ingångar till fortsatt forskning inom området att diskuteras. Avslutningsvis sammanfattas studiens slutsats utifrån frågeställningen.

5.1 Jämförelse av läromedel

Utifrån resultatet som helhet tycker jag mig se en skiljelinje mellan två typer av läromedel. Den första typen har ett tydligare fokus på tekniska aspekter av programmeringen, och den andra fokuserar istället på användning av programmering och problemlösning.

Den första av dessa kopplar jag till läromedlen Numerus och Matematik 5000+. Här återfinns en stor andel frågor av teknisk karaktär, få problemlösningsuppgifter och uppgifterna är överlag också bakåtsyftande. I den instrumentella genesisen finns här också bara möjlighet för eleverna att ta några första begränsande steg, som berör gester och upptäckande och selektion.

Utifrån resultatet är det lätt att koppla en sådan beskrivning av ett läromedel till Numerus, men Matematik 5000+ avviker på vissa punkter. Exempelvis berörs en generell teknik och någon av de specifika teknikerna. Jag vill dock hävda att Matematik 5000+ trots detta ger ett liknande helhetsuttryck då de uppgifter som eleverna faktiskt får arbeta med inte innehåller problemlösning och till hög grad är bakåtsyftande. Så även om läromedlet innehåller aspekter som är mer riktade mot problemlösning ges eleverna aldrig möjlighet att själva angripa problemlösningsuppgifter. Inom denna grupp finns också en skillnad i hur tekniska aspekter tas upp, där Numerus behandlar innehållet explicit på ett helt annat sätt. Här återfinns förklaringar och beskrivningar till de koder och kommandon som används, medan de i Matematik 5000+ endast visas i exempel.

Helhetsintrycket av Exponent och Origo är istället ett fokus på användningen av programmering och problemlösning, alltså den andra typen av läromedel. Här återfinns istället en större andel problemlösningsuppgifter och ett tydligare fokus på de senare delarna av den instrumentella genesisen, som instrumentella handlingsscheman och tekniker, samt selektion och personalisering. Uppgifterna är också aktivitetsstödjande eller problemorienterade till sin karaktär.

Gemensamt för båda dessa läromedel är att mer tekniska, grundläggande aspekter inte förklaras och beskrivs explicit, eller inte tas upp alls. Gester och upptäckt av artefakten ges alltså ett väldigt begränsat stöd, trots att dessa delar sedan krävs i uppgifterna. En skillnad finns också mellan läromedlen, där Exponent innehåller enstaka exempel och sedan uppgifter som eleverna får angripa på egen hand. Dessa uppgifter är också problemlösningsuppgifter till hög grad. I Origo är läromedlet istället organiserat i aktiviteter som innehåller problemlösning. Delfrågor ger eleverna här stöd längs vägen och som en följd av detta är uppgifterna inte problemlösning till lika hög grad.

5.2 Konsekvenser för lärare

För att en värdering av de olika läromedlen ska vara möjligt behöver de sättas i de sammanhang där de är tänkta att användas: I händerna på elever i en klass, och som en del i ett arbete som ska följa aktuella styrdokument. Utifrån kursplanens beskrivning av programmering som en strategi för matematisk problemlösning (Skolverket, 2017a) blir det intressant att lyfta fram de delar som läromedlen bidrar med, och det som läraren behöver tillföra utöver materialet som är givet.

För läromedel med ett mer tekniskt fokus, Numerus och Matematik 5000+, saknas innehåll som uppfyller styrdokumentens beskrivning av programmering och problemlösning. Här krävs alltså att läraren, utöver innehållet i läromedlet, genom aktiviteter eller extra uppgifter, faktisk använder programmering för problemlösning, eftersom inte detta görs i läromedlet.

De läromedel med större fokus på användning av programmering och problemlösning, Exponent och Origo, kommer istället uppfylla styrdokumentens beskrivning mer direkt. Utifrån analysen som är gjord är det dock tydligt att också dessa läromedel saknar delar. Här är det istället de tidigare stegen i den instrumentella genesisen som läraren måste stödja eleverna i, eftersom detta stöd till stor del saknas i läromedlen.

Utifrån Johanssons (2005) beskrivning av hur läromedel påverkar undervisningen kan vi sedan ifrågasätta hur rimligt det är att en lärare faktiskt kompletterar läromedlet på det sätt som beskrivits ovan. Johansson menar att lärare oftast inte presenterar innehåll som inte inkluderas, vilket gör det troligt att lärare inte kommer lägga till programmering för problemlösning om dessa delar inte finns med i läromedlet. Läromedlen med fokus på användning av programmering och problemlösning saknar inte något innehåll på samma sätt, utan skulle istället behöva en tydligare behandling av grunderna. Eftersom dessa grunder behövs i nästa steg kommer de dock inte kunna tappas bort på samma sätt, utan måste ändå behandlas, om inte annat så när elever fastnar och ställer frågor medan de arbetar med innehållet. Jag vill därför hävda att bristerna i läromedlen med ett tekniskt fokus riskerar att påverka undervisningen genom dessa brister mer än de läromedel som har fokus på användning av programmering och problemlösning. Hur mycket av de tekniska aspekterna som faktiskt behövs ta upp är också svårt att avgöra på ett generellt plan, då eleverna ska börja med programmeringen under grundskolan.

Sammantaget visar dock analysen att lärare, oavsett läromedel, behöver förhålla sig självständigt till det material som används. Inget läromedel fångar alla delar av styrdokumentens beskrivning, och de delar som den instrumentella ansatsen visar behövs för att nå dit. Den instrumentella orkestreringen pekar också på detta genom punkten det didaktiska utförandet. En lärare behöver planera användandet av ett läromedel, men även om läromedlet följs så nära som möjligt kommer läraren ändå behöva agera självständigt i relationen till eleverna i klassrummet när ett läromedel används. Hur läraren bemöter frågor som dyker upp, lärarens inställning till aktuellt innehåll och vilken hänsyn som tas till elevernas förkunskaper kommer alla spela roll i hur innehållet i ett läromedel faktiskt förmedlas vidare.

Lärarens betydelse ser vi också i utifrån till Kilhamn och Bråting (*kommande*) och Misfeldt och Ejsing-Dunn (2015). De lyfter båda fram hur läraren hanterar programmeringen och det aktuella innehållet som avgörande för att lyfta den matematiska kopplingen i arbetet, och därigenom komma åt de möjligheter som tillskrivs programmering i matematiken. Det matematiska innehållet behöver alltså lyftas fram, och relationen mellan matematik och programmering,

gällande begrepp och symboler, behöver göras explicit för elever. Denna aspekt saknas i stort hos alla läromedel, och endast Numerus berör delar av den. Det är alltså något som läraren måste bidra med.

5.3 Programmering som verktyg

Den analys som har genomförts har också givit vissa insikter i programmering som verktyg för problemlösning. Den instrumentella ansatsen slår fast skillnaden mellan artefakt och instrument. Programmeringen blir utifrån detta inte ett verktyg för problemlösning för än ett instrument byggs upp genom en process av instrumentell genesis. Denna process innebär vidare en instrumentering, utveckling av scheman, och en instrumentalisering, formande av artefakten.

Instrumenteringen sker på flera nivåer. Den första är utveckling av användningsscheman, vilket tar sig uttryck i gester för att hantera artefakten. Det handlar här om syntax och olika typer av kommandon och dess funktioner. Den andra är instrumentella handlingsscheman, vilka tar sig uttryck i tekniker riktade mot problemlösning. Här har identifierats såväl generella som specifika tekniker. Generella tekniker handlar då om hur problemlösning kan angripas med hjälp av programmering. I materialet har dessa fokuserat på vikten av att beskriva ett programs funktion och struktur innan själva kodningen börjar. Specifika tekniker blir istället angreppsätt för olika typer av problem. De tre specifika teknikerna att simulera slumphändelser, pröva med villkor och iterativt generera har identifierats i materialet.

Instrumentaliseringen kan sedan beskrivas i fyra steg. Från upptäckande och selektion, via personalisering till transformering av artefakten. Stöd och utrymme av de tre första av dessa har identifierats i materialet. Dessa steg sker genom att eleven får möta kod och kommandon, välja ut och kombinera dessa och slutligen egen formulering av en lösning till ett problem, med både övergripande struktur och den kod som behövs för att utföra lösningen. Den sista punkten, transformering, har inte identifierats i materialet men skulle kunna komma till uttryck när eleven själv formar artefakten genom att välja nödvändiga paket och importera dessa för att få tillgång till de kommandon som behövs för det aktuella problemet.

Som helhet blir det tydligt att många olika aspekter finns till den process där programmering ska bli ett verktyg för problemlösning i elevernas händer. Det blir också tydligt att en undervisning som syftar till detta måste jobba brett. Eleverna behöver få möta de mer tekniska aspekterna, bekanta sig med syntax och förstå funktionen hos kommandon. De behöver stöd i hur programmering sedan kan användas för problemlösning, förslagsvis genom exempel eller stödjande frågor. De behöver också få möta generella och specifika tekniker för problemlösning. Slutligen behöver de också få möta problemlösning på egen hand och pröva det verktyg de har för att kunna bekanta sig med det ytterligare och identifiera sina egna utvecklingsområden.

Utöver denna självständiga användning, som framförallt är en del av Exponent, finns också några områden som saknas i alla läromedel. Instrumentaliseringen innehåller i ett lite större perspektiv också aspekter av när programmering ska användas som verktyg för problemlösning. Genom exempel kan eleverna få en bild av vad programmering kan användas till men frågan om vad programmeringen kan bidra med i problemlösning tas egentligen inte upp av något läromedel. Eleverna behöver heller aldrig välja verktyg själv, utan det framgår alltid av sammanhanget när programmering är det verktyg som ska användas. För att ytterligare utveckla elevernas problemlösning med programmering vill jag hävda att de också behöver stå i situationer där flera olika verktyg är möjliga. Här behöver de få utrymme att värdera de olika

verktygen och välja ett utifrån situation och det som efterfrågas, eller kombinera verktygen och låta dem bidra på de områden de är lämpade.

5.4 Metod

Jag vill också lyfta några frågor kring den metod som används i studien, både gällande urval och analys. En viktig första aspekt är det material som studerats. Urvalet har begränsats till läromedel i form av klassiska läroböcker, med tillhörande extramaterial. Urvalet inom dessa bedöms vara i princip heltäckande, men den avgränsning som används kan ifrågasättas. Digitala läromedel, där programmering kan integreras i lärmiljön på ett annat sätt, har helt uteslutits från studien. Förekomsten av denna typ av läromedel är svår att bedöma, men mitt intryck är att det används mer i undantagsfall. Det är dock mycket möjligt att det blir allt vanligare, speciellt i samband med programmering. Möjligheterna i en sådan implementering av en programmeringsmiljö som är integrerad med elevernas läromedel är alltså fortfarande outforskade. Material där endast programmering i matematik behandlas saknas också i studien. Vilka sådana material som används, och hur vanligt förekommande de är, är tyvärr svårt att uttala sig om. Det är möjligt att lärare behöver hitta egna komplement till vanliga läromedel eftersom uppdatering av heltäckande läromedel tar tid, och inköp av nya läromedel för mindre ändringar i styrdokument kan ses som kostsamt. Denna typ av läromedel är alltså också outforskade.

En annan aspekt är det perspektiv som används för analysen. Samma ramverk hade också kunnat möjliggöra andra perspektiv, genom andra val av artefakt och subjekt. För denna studie har matematisk problemlösning setts som objekt. Detta ger en stor, övergripande bild på problemlösning, och den instrumentella processen blir därmed något utdraget som kan ske under en, eller flera matematikkurser. Det större perspektivet gör det dock svårare att lyfta fram vad som händer och krävs av en elev i en enskild problemlösning. Väljs ett specifikt problem som objekt hade kanske detta kunnat lyftas fram på ett tydligare sätt. Ett annat typ av material hade behövts för en sådan analys, och frågeställningen hade blivit en annan. Valt perspektiv gjorde det alltså möjligt att studera läromedlen som helhet, och ligger på så sätt närmre frågeställningen.

En instrumentella ansats till programmering tycks också vara ett område under utveckling. Valet av artefakt och instrument kan också diskuteras utifrån den situation som studeras, och vad syftet med studien är. Den ingång som används här ser jag därför inget problem med, men jag vill öppna för att vidare insikter i området kan lyfta fram styrkor med andra perspektiv. En djupare analys av programmering ur en instrumentell ansats skulle därför behövas, och hade kanske varit en tillgång och gett studien en tydligare teoretisk förankring.

5.5 Vidare forskning

Även om kartläggning av hur programmering används i läromedel kan tyckas tillräckligt i och med denna studie är det tydligt att många frågor kvarstår angående programmering i matematiken. Jag ser här en brist i vår förståelse, både gällande processen av problemlösning i matematik med hjälp av programmering i sig, men också en tydligare bild av vad elever egentligen kan förväntas lära sig när det gäller programmering som verktyg för problemlösning. Två möjliga ingångar för att studera dessa frågor vidare har blivit synliga under arbetet med denna studie.

En första ingång är att studera problemlösningsprocessen i sig utifrån en instrumentell ansats. Utifrån ett programmeringsspråk som artefakt, och lösningen av ett specifikt problem som

objekt, blir då skapandet av en programkod, samt scheman för hur denna används och tolkas, den instrumentella genesisen. Programkoden och kunskapen om hur denna ska användas blir då resultatet, instrumentet som eleven kan använda för att lösa det aktuella problemet. Den teoretiska ram som den instrumentella ansatsen ger för att förstå den instrumentella genesisen kan då användas för att lyfta fram och belysa de olika komponenter som ingår i problemlösningsprocessen.

Utifrån en sådan analys av elevers problemlösningsprocess, och de krav denna process ställer på eleverna, kan det sedan bli möjligt att utforma stöd för eleverna i detta. Det kan också belysa förkunskaper och kompetenser som eleverna behöver utveckla inför arbetet med programmering som verktyg för problemlösning.

En andra ingång är att bygga vidare på den ansats som använts i denna studie och fokusera på de tekniker som kan utvecklas av elever. Såväl generella som specifika tekniker kan då vara av intresse. Skulle en formulering av en generell teknik kunna visas stödja elever i problemlösning med programmering skulle det vara intressant. Vidare kan de tre specifika tekniker som identifierades här, att simulera slumphändelser, pröva med villkor och iterativt generera, kan ses som en ingång till möjliga specifika tekniker. Ett bredare material kan synliggöra flera tekniker att lägga till i listan ovan, eller ge andra perspektiv på vad som uppgifterna kräver, och därmed ge anledning till att förändra kategorierna.

En sådan ansats kan tydliggöra de möjligheter som finns i att arbeta med programmering för problemlösning, och visa på olika områden där kopplingen till matematik gör uppgifter intressanta att angripa för elever. Med en samling av tekniker tillgängliga som syftar just till problemlösning kan det också bli lättare för lärare att rikta programmeringen mot just problemlösningen. Som denna studie har visat är det inte självklart att läromedlet som används lägger fokus på samma områden som styrdokumenten.

5.6 Slutsats

Slutligen vill jag återkoppla denna diskussion till studiens frågeställning:

Hur och till vilken grad ges eleverna stöd och möjlighet att tillägna sig programmering som ett eget verktyg att hantera för problemlösning i matematik genom läromedel för gymnasiet?

Den mer beskrivande delen av hur stöd och möjligheterna ser ut är svår att sammanfatta i en enklare slutsats. För en beskrivning av varje läromedel och de olika aspekterna av detta hänvisas till resultatet. Det är dock tydligt att läromedlen valt olika fokus på programmeringen, och utifrån analysen har också identifierat skillnader i graden av stöd. De två läromedlen Exponent och Origo är de som faktiskt har behandlat problemlösning och de gör detta genom exempel och problemorienterade uppgifter respektive större aktiviteter med stödjande frågor. Genom att möta generella och specifika tekniker får eleverna ingångar till hur programmering kan användas för problemlösning. Utrymme finns här också, framförallt i Exponent, för eleverna att möta problemsituationer på egen hand och genom detta göra programmeringen allt mer till ett eget verktyg. Båda dessa bister dock i sitt stöd kring mer grundläggande tekniska aspekter.

Numerus har ett tydlig fokus på tekniska aspekter, och behandlar också detta innehåll explicit. Problemlösning saknas, men den explicita behandlingen leder till att viktiga frågor om begreppsanvändning inom matematik och programmering, exempelvis variabel, lyfts fram. Avsaknaden av problemlösning gör det dock tydligt att Numerus behöver kompletteras med

annat innehåll för att styrdokumentens skrivning om programmering och problemlösning ska kunna uppfyllas.

Matematik 5000+ har också ett fokus på tekniska aspekter, även om sammanhanget är annorlunda än Exponent. Här finns beskrivningar av programmering för problemlösning, men ingen problemlösning återfinns i uppgifterna. Till skillnad från Numerus behandlas inte heller de tekniska aspekterna explicit, vilket gör att de frågor som kan lyftas i samband med de tekniska aspekterna inte tas upp.

Vidare har den instrumentella ansatsen visat att tillägna sig programmering som ett verktyg för problemlösning är en process med många olika komponenter. Detta kan alltså inte betraktas som något självklart, utan elever behöver stöd och goda förutsättningar för denna process. Det är också en process som kommer att ta tid, också för elever som redan har grundläggande förkunskaper inom programmering med sig från grundskolan.

Oavsett läromedel behöver vi som lärare därför vara medvetna om vad det är vi vill med vår undervisning. Det är en resa vi tar med eleverna på, och som lärare behöver vi veta vart vi är på väg, vilka steg som behövs på vägen och de svårigheter som finns. Vi behöver därför en förståelse för programmeringens möjligheter och begränsningar, syftet med den och den process det är att tillägna sig programmeringen som ett verktyg. I denna studie har delar av detta kunnat beröras, men det är tydligt att mycket arbete finns kvar att göra. Ett arbete som behöver utföras av såväl forskare som lärare. Vi behöver en djupare förståelse av programmeringens möjligheter och begränsningar, samt den process som problemlösning med programmering innebär. Vi behöver också mer exempel på hur programmeringen kan användas på ett fruktbart sätt inom matematiken, såväl praktiska lösningar som valet av matematiskt innehåll. Vi behöver alltså fortsätta utforska och våga utveckla vår undervisning. Först då kan programmeringen bli intressant och utmanande för elever, och inte bara en punkt i styrdokumenten att bocka av.

6 Referenslista

Alfredsson, L., Heikne, H., Holmström, B., Dyrander, J., & Karlsson, M. (2018). *Matematik 5000 Kurs 1c.* Stockholm: Natur & Kultur.

Blackwell, A. (2002). What is Programming? In J. Kuljis, L. Baldwin & R. Scoble (Ed). *Proc. PPIG 14* (pp. 204-218).

Brehmer, D., Ryve, A., & Van Steenbrugge, H. (2016). Problem solving in Swedish mathematics textbooks for upper secondary school. *Scandinavian Journal of Educational Research, 60*(6), 577-593.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77-101.

Drijvers, P., Kieran, C., Mariotti, M., Ainley, J., Andresen, M., Chan, Y., . . . Lagrange, J. (2010). Integrating Technology into Mathematics Education: Theoretical Perspectives. In C., Hoyles, & J.B., Lagrange (Ed) *Mathematics Education and Technology-Rethinking the Terrain: The 17th ICMI Study* (Vol. 13, New ICMI Study Series, pp. 89-132). Boston, MA: Springer US.

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

Gleerups(*u.å.*a). *Exponent – programmering och matematisk problemlösning – Exempel*. Hämtad 2019-04-10 från https://gleerupsportal.se/laromedel/publik-artikel/2c46089d-1a0d-4320-900f-4bde2839b337?_ga=2.250725635.406599843.1557753827-1514766068.1554450341

Gleerups(*u.å.*b). *Exponent – programmering och matematisk problemlösning – Övningar*. Hämtad 2019-04-10 från https://gleerupsportal.se/laromedel/publik-artikel/83b119c4-ceea-4d6f-90fc-b361e3b91079?_ga=2.250725635.406599843.1557753827-1514766068.1554450341

Guin, D., & Trouche, L. (1998). The Complex Process of Converting Tools into Mathematical Instruments: The Case of Calculators. *International Journal of Computers for Mathematical Learning, 3*(3), 195-227.

Klilhamn, C. & Bråting, K. (*kommande*). Algebraic thinking in the shadow of programming. To be published in U. T. Jankvist, M. van den Heuvel-Panhuizen, & M. Veldhuis (Eds.), *Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education*, (pp. xxxx-yyyy). Utrecht, the Netherlands: Freudenthal Group & Freudenthal Institute, Utrecht University and ERME.

Lithner, J. (2006). A framework for analysing creative and imitative mathematical reasoning *(Department of Mathematics and Mathematical Statistics, Umeå University, Research Reports in Mathematics Education, No. 2)*. Umeå: Umeå universitet.

Lithner, J. (2008). A research framework for creative and imitative reasoning. *Educational Studies in Mathematics, 67* (3), 255–276.

Johansson, M. (2003). *Textbooks in mathematics education: A study of textbooks as the potentially implemented curriculum*. Licentiate thesis / Luleå University of Technology, 2003.

Johansson, M. (2005*)*. The Mathematics Textbook. From artefact to instrument. *Nordic Studies in Mathematics Education No 3-4*,

Misfeldt, M. & Ejsing-Duun, S. (2015). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. *CERME 9 - Ninth Congress of the European Society for Research in Mathematics Education*. Charles University in Prague, Faculty of Education. pp.2524-2530.

Mullis, I., Martin, M. & Foy, P. (2008*). TIMSS 2007 International Mathematics Report: Findings from IEA´s Trends in International Mathematics and Science Study at the Fourth and Eight Grades.* Boston: TIMSS & PIRLS International Study Centre.

Pepin, B., & Haggarty, L. (2001). Mathematics textbooks and their use in English, French and German classrooms:. *Zentralblatt Für Didaktik Der Mathematik, 33*(5), 158-175.

Rabardel, & Bourmaud. (2003). From computer to instrument system: A developmental perspective. *Interacting with Computers, 15*(5), 665-691.

Rolandsson, L., & Skogh, I. (2014). Programming in School: Look Back to Move Forward. *ACM Transactions on Computing Education (TOCE)*, 14(2), 1-25.

Rung, A., Von Heijne, E., & Rundlöf, T. (2018). *Matematik Numerus 1c*. Stockholm: Liber.

Sanoma(*u.å.*). *Origo – programmering.* Hämtad 2019-04-10 från https://www.sanomautbildning.se/globalassets/ak-7-9/rakna-med-kod/matematikorigoprogrammeringsaktiviteter_kurs-1-5.pdf

Schoenfeld, A. (1985). *Mathematical Problem Solving*. Orlando, FL: Academic Press.

Schoenfeld, A. (2016). Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics (reprint). *Journal of Education, 196*(2), 1-38.

Skolverket. (2017a*). Ämne – Matematik för gymnasieskolan*. Hämtad 2019-05-06 från https://www.skolverket.se/undervisning/gymnasieskolan/laroplan-program-och-amnen-i-gymnasieskolan/gymnasieprogrammen/amne?url=1530314731%2Fsyllabuscw%2Fjsp%2Fsubject.htm%3FsubjectCode%3DMAT%26tos%3Dgy&sv.url=12.5dfee44715d35a5cdfa92a3

Skolverket. (2017b). *Kommentarmaterial till ämnesplanen i matematik i gymnasieskolan.* Hämtad 2019-05-06 från https://www.skolverket.se/download/18.6011fe501629fd150a2893a/1530187438471/Kommentarmaterial_gymnasieskolan_matematik.pdf

Skolverket. (2017c). *Kommentarmaterial till kursplanen i matematik*. Hämtad 2019-05-06 från https://www.skolverket.se/publikationsserier/kommentarmaterial/2017/kommentarmaterial-till-kursplanen-i-matematik-reviderad-2017?id=3794

Skolverket. (2018). *Kursplan – Matematik.* Hämtad 2019-05-06 från https://www.skolverket.se/undervisning/grundskolan/laroplan-och-kursplaner-for-grundskolan/laroplan-lgr11-for-grundskolan-samt-for-forskoleklassen-och-fritidshemmet?url=1530314731%2Fcompulsorycw%2Fjsp%2Fsubject.htm%3FsubjectCode%3DGRGRMAT01%26tos%3Dgr&sv.url=12.5dfee44715d35a5cdfa219f

Trouche, L. (2005). An Instrumental Approach to Mathematics Learning in Symbolic Calculator Environments. In Trouche, L., Guin, D., & Ruthven, K. (Ed) *The Didactical Challenge of Symbolic Calculators: Turning a Computational Device into a Mathematical Instrument* (Vol. 36, Mathematics Education Library, pp. 137-162). Boston, MA: Springer US.

Trouche, L., & Drijvers, P. (2014). Webbing and orchestration. Two interrelated views on digital tools in mathematics education. *Teaching Mathematics and Its Applications: An International Journal of the IMA, 33*(3), 193-209.

Verillon, P., & Rabardel, P. (1995). Cognition and artifacts: A contribution to the study of though in relation to instrumented activity. *European Journal of Psychology of Education, 10*(1), 77-101.

Vetenskapsrådet (2002). Forskningsetiska principer inom humanistisk-samhällsvetenskaplig forskning. Stockholm: Vetenskapsrådet.