Literatuurreview uitgevoerd in opdracht van de Kennisrotonde, het online loket voor de beantwoording van actuele kennisvragen uit het onderwijs.

Leren programmeren in het PO – een literatuurreview –

Johan Jeuring, Gemma Corbalan, Nienke van Es, Hanneke Leeuwestein

Universiteit Utrecht

Joek van Montfort

Xota

Juli 2016

1 Introductie

In sommige Europese landen, zoals Estland en Groot-Britannië, zijn flinke stappen zijn gezet in het programmeeronderwijs voor kinderen in de basisschoolleeftijd (Brown et al. [2016]). De Tweede Kamer heeft in 2014 een motie aangenomen die de staatssecretaris vraagt te onderzoeken op welke wijze op Nederlandse basisscholen aandacht kan worden besteed aan digitale geletterdheid en programmeren. Een aantal basisscholen heeft al enige ervaring opgedaan met programmeeronderwijs, en recent heeft de PO-Raad een leerlijn programmeren gelanceerd. Bij al deze initiatieven komt de vraag hoe je het best programmeeronderwijs kunt verzorgen natuurlijk naar boven. Welke aanpakken laten goede resultaten zien? Is er verschil in motivatie en resultaat tussen jongens en meisjes? Moet je kinderen in de onderbouw van het primair onderwijs ander programmeeronderwijs aanbieden dan kinderen in de bovenbouw? Deze vragen kwamen ook in de Kennisrotonde van het Nationaal Regieorgaan Onderwijsonderzoek (NRO) naar boven, en het onderzoek beschreven in dit rapport komt voort uit die vragen.

2 Wat is programmeren?

Wat is programmeren? Dit is een vraag die niet iedereen op dezelfde manier beantwoordt.

Er bestaan verschillende definities van het begrip programmeren. In dit rapport hanteren we de volgende definitie. Programmeren is het opstellen van een programma: een artefact dat door een machine, robot, of een tool geïnterpreteerd of na vertaling geëxecuteerd kan worden, of zelfs zonder machine uitgevoerd kan worden. Vaak bestaat een programma uit een lijst van instructies, maar een programma kan ook hele andere vormen aannemen, zoals een functie, of zelfs een machinebeschrijving. Fundamentele onderdelen van programma's zijn onder andere volgorde (doe iets nadat iets anders is gedaan), keuze (voer het één of het ander uit afhankelijk van of een conditie waar is of niet), herhaling (doe iets net zo vaak totdat aan een bepaalde conditie is voldaan), toestandsvariabele (om een waarde in op te slaan), en functie (neem een argument en lever een resultaat op). Programmeren kan gedaan worden in een programmeertaal zoals Java, C, Python, Scheme, of Haskell (met vaak weer een specifieke, visuele, op onderwijs gerichte omgeving, zoals Greenfoot, Alice, BlueJ, DrRacket), maar het kan ook gedaan worden in een tool of een game, zoals Scratch, Lightbot, Move the Turtle, de website `code.org`, etc. Ook bestaan er *unplugged* methoden om programmeren te leren, dwz. methoden waarbij in het geheel geen gebruik van computers wordt gemaakt. In de verschillende experimenten in het PO komen veel aspecten van het programmeren aan de orde, maar zal minder of geen aandacht besteed worden aan abstractiemechanismen zoals methodes, klassehiërarchieën of modules, en zal de grootte van programma's beperkt blijven. Het schrijven van webpagina's

```
#main program starts here
playAgain = 'yes'
while playAgain == 'yes' or playAgain == 'y':

    scene1()

    #use the makeChoice function to get
    #the player to decide which way to go
    firstChoice  = makeChoice()

    #this if else statement will show the next
    #scene based on the players choice
    if firstChoice == '1':
        scene2A()
    else:
        scene2B()

    secondChoice = makeChoice()

    #this if else statement will show the next
    #scene based on the players choice
    if secondChoice == '1':
        goodEnd()
    else:
        badEnd()

    print('Do you want to play again? (yes or no)')
    playAgain = input()
```

Figuur 1: Een programma in tekstuele en visuele vorm, zie:
`https://cdathenry.wordpress.com/2013/11/09/`

(html), en het maken van *powerpoints*, filmpjes, of andere media vallen in het algemeen niet onder programmeren.

Programma's die kinderen in het PO *schrijven* hebben vaak één van de volgende vier vormen: tekstueel, visueel, unplugged, of (fysieke) blokken. We beschrijven ieder van de vormen kort.

2.1 Tekstueel programmeren (verschillende programmeertalen)

Dit is de traditionele manier van programmeren. Experts programmeren in tekstuele programmeertalen zoals Java, C, Python, Scheme of Haskell. Om een tekstueel programma te schrijven moet een leerling de grammatica van de programmeertaal kennen en de namen van de commando's die specifieke taken uitvoeren (zie Figuur 1 links).

2.2 Visueel programmeren (bv. Scratch, Lego MindStorms)

In een visuele programmeeromgeving stelt een leerling een programma samen door blokken te slepen en aan elkaar te klikken. Hierdoor hoeft ze niet eerst de grammatica van een programmeertaal te leren, maar kan ze al snel aan de slag met het maken van bijvoorbeeld een game. Uiteraard heeft ook een visueel programma een onderliggende grammatica, maar de vorm van de blokken zorgt er voor dat het moeilijk of vaak onmogelijk is om een grammaticale fout

Figuur 2: Leerlingen lopen over een sorteernetwerk, zie:
`http://csunplugged.org/sorting-networks/`

te maken. De blokstructuur helpt bij het snel ontwikkelen van correcte programma's (zie Figuur 1 rechts).

2.3 Unplugged programmeren

Bij unplugged programmeren maakt een leerling geen gebruik van een computer, maar schrijft bijvoorbeeld een programma door een aantal kaarten met instructies ('ga drie stappen vooruit', 'draai 90 graden naar rechts') daarop achter elkaar te leggen, of door een andere leerling heel precies verbaal te instrueren over hoe zij moet bewegen. De term 'unplugged' geeft aan dat er geen elektriciteit wordt gebruikt bij deze programmeeractiviteit. Figuur 2 laat bijvoorbeeld zien hoe leerlingen een rij getallen sorteren door over een zogenaamd sorteernetwerk te lopen.

2.4 Programmeren met behulp van fysieke blokken

Een specifieke manier van unplugged programmeren is het programmeren met fysieke blokken, zie Figuur 3. Vooral bij jongere leerlingen wordt veel gebruik gemaakt van blokken: voor deze review hebben we 10 artikelen gevonden waarin experimenten met blokken worden uitgevoerd. Een blok representeert een commando. Naast een commando staat er op een blok vaak ook een streepjescode, of een andere representatie van het commando, die kan worden ingescand, en vervolgens op een computer, of door een robot, of door een bak met lampjes er in, uitgevoerd.

Om te zien wat een programma doet wordt een programma uitgevoerd. Net als bij het schrijven van een programma kan het resultaat van het uitvoeren van een programma verschillende vormen aannemen:

- grafisch (vaak mensen of beesten die in een virtuele wereld bewegen, zoals bijvoorbeeld in Scratch, bij `code.org`, HopScotch, enz.);
- tekstueel (verhalend, chatbox);
- bewegende robots (Lego Mindstorms);

Figuur 3: Blokken om mee te programmeren (Sullivan and Bers [2016])

- unplugged (instructies op papier door kinderen uit laten voeren).

3 Onderzoeksvraag en -methode

De onderzoeksvraag vanuit de Kennisrotonde luidt:

> Wat weten we over de effecten van programmeeronderwijs op programmeervaardigheden van leerlingen tot 12 jaar?

Om de onderzoeksvraag te beantwoorden voeren we een literatuurreview uit. Bij deze review hebben we de volgende deelvragen meegenomen:

- Welke vormen van programmeeronderwijs zijn gangbaar?
- Wat is bekend over de opbrengsten van dit programmeeronderwijs?
- Zijn opbrengsten verschillend voor verschillende leerlingen (jongens versus meisjes, sociaaleconomische status, voorkennis, interesse, leeftijd)?

Kern van de review is de vraag of er mechanismen zijn te benoemen die verklaren waarom bepaalde vormen van programmeeronderwijs al dan niet succesvol zijn voor bepaalde groepen leerlingen. Inzicht in deze mechanismen kunnen leraren en bestuurders helpen om keuzes te maken voor het programmeeronderwijs.

3.1 Literatuur

Voor het vinden van voor deze review relevante artikelen voeren we een bibliografische zoekprocedure uit.

We zoeken naar relevante artikelen in de artikelendatabases Scopus (een algemene database), ACM Digital Library (ACM DL, een database die vooral veel artikelen over allerlei aspecten rondom informatica bevat), en ERIC (een database die vooral veel artikelen over onderwijskundig onderzoek bevat).

Daarnaast zoeken we naar eerdere reviews over de effecten van programmeeronderwijs op programmeervaardigheden van leerlingen tot 12 jaar.

3.2 Zoektermen

Om artikelen te vinden hebben we verschillende zoektermen gebruikt. Op de ACM DL zijn we bijvoorbeeld begonnen met de volgende zoekterm:

> (programming OR coding OR computer science
> OR computational thinking)
> AND
> (primary school OR elementary school OR kids
> OR children OR young people OR youth OR students)
> AND
> (learning OR teaching)

Dit leverde 17,705 resultaten op; te veel om voor nadere selectie te gebruiken. Door het weghalen van de term *students* krijgen we 508 resultaten. Voor de verschillende databases hebben we de zoekterm echter moeten beperken of wijzigen. Uiteraard bevatten al onze zoektermen onderdelen van de bovenstaande zoekterm.

3.3 Selectiecriteria

Voor het selecteren van artikelen gebruiken we de volgende inclusiecriteria:

- Inhoud: een artikel beschrijft een interventie in het programmeeronderwijs. Het beschrijft een tool, omgeving, game, of methode voor programmeeronderwijs die daadwerkelijk in het primair onderwijs wordt ingezet.
- Leeftijd: gericht op het primair onderwijs. Omdat andere landen vaak andere schoolsystemen hanteren nemen we artikelen mee als de onderzochte leeftijdsgroep overlapt met de leeftijd 6-12 jaar.
- Doelgroep: algemeen. Artikelen gericht op doelgroepen met speciale leerbehoeften, zoals dove kinderen, of kinderen die vanwege een handicap niet kunnen typen, nemen we niet mee.
- Publicatievorm: tijdschriftartikelen, artikelen in proceedings, hoofdstukken in boeken, mits peer-reviewed. *Full text* is beschikbaar. Technische rapporten nemen we niet mee.
- Lengte: een artikel is minimaal 4 pagina's lang. In kortere artikelen is de informatie vaak te beperkt om conclusies te kunnen trekken of controleren.
- Publicatiedatum: na 2004 gepubliceerd. In uitzonderingsgevallen, zoals wanneer een interventie wordt beschreven die ook nu nog gemakkelijk uitgevoerd zou kunnen worden, wijken we hier van af.
- Onderzoeksopzet: randomized controlled trials, maar ook goede kwalitatieve studies, zoals bijv. case studies, met objectieve uitkomstmaten.

Bij twijfel of een artikel voldoet aan de selectiecriteria wordt het artikel meegenomen, en stelt een tweede persoon in de coderingsfase vast of het artikel wel

of niet moet worden meegenomen, eventueel in overleg met de persoon die het artikel in eerste instantie geselecteerd heeft. In sommige gevallen hebben zelfs in totaal drie of vier personen nagedacht over of een artikel wel of niet moet worden meegenomen.

3.4 Artikelenselectie

De verschillende zoekacties leveren de volgende resultaten op.

We hebben een aantal gerelateerde reviews gevonden (Yurdugül and Aşkar [2013], Lye and Koh [2014], Grover and Pea [2013], Moreno-León and Robles [2016]), maar geen review kijkt specifiek naar het effect van programmeeronderwijs op programmeervaardigheden. De gevonden reviews hebben vaak als doel om het effect van programmeeronderwijs op juist andere vaardigheden dan programmeren te bestuderen, zoals probleemoplossend vermogen, computational skills, of vaardigheden in andere domeinen.

In de ACM DL hebben we 508 artikelen gevonden met behulp van de zoekterm uit sectie 3.2. Van deze 508 artikelen heeft één van de auteurs 52 artikelen geselecteerd.

In ERIC hebben we met behulp van de zoekterm

> programming AND (learning OR teaching)

en daarna filteren op

> elementary education and only peer-reviewed studies

135 artikelen gevonden. Van deze 135 artikelen zijn 17 geselecteerd.

Via Scopus, met varianten van de zoekterm uit sectie 3.2, hebben we 70 artikelen gevonden, waarvan 6 zijn geselecteerd.

Naast bovenstaande zoekacties in wetenschappelijk databases hebben we ook artikelen proberen te vinden door experts te vragen, webpagina's van bekende onderzoekers in het veld te bezoeken, webresources van lerarenverenigingen in verschillende landen te bekijken, enz. Op deze manier hebben we nog 10 artikelen gevonden.

In totaal hebben we 85 artikelen geselecteerd. Van 5 artikelen hebben we in de ons beschikbare tijd geen full text kunnen vinden. In de coderingsfase zijn van de overgebleven 80 artikelen nog eens 32 artikelen afgevallen, omdat ze toch niet aan één of meerdere selectiecriteria voldoen.

4 Codering artikelen

We hebben alle geselecteerde artikelen gecodeerd om de volgende gegevens te verzamelen:

- publicatie: waar en wanneer gepubliceerd
- interventie: hoe programmeren leerlingen (met welke programmeertaal, programmeeromgeving, programmeermethode), welke activiteiten voeren leerlingen uit, duur, begeleiding
- deelnemers: het aantal deelnemers aan de interventie, hoe ze geselecteerd zijn, leeftijd, geslacht, nationaliteit
- resultaten: uitkomstmaat, resultaten, conclusies

Soms ontbreken beschrijvingen van een gedeelte van deze gegevens.

De resultaten van de coderingen hebben we verzameld in een spreadsheet.

5 Resultaten

5.1 Typering van de gevonden literatuur

De geselecteerde artikelen zijn vooral afkomstig uit de VS, maar we hebben ook artikelen uit Oostenrijk, Bulgarije, China, Duitsland, Griekenland, Slowakije, Spanje, Switzerland, Taiwan en Turkije geselecteerd. De geselecteerde artikelen beschrijven onderwijsinterventies uitgevoerd tussen 2005 en 2016 in de leeftijdsgroep van het primair onderwijs, maar we hebben ook nog drie artikelen van voor 2005 geselecteerd. De onderwijsinterventies duren tussen de één (of meerdere) sessies van ongeveer 2 uur en 3 jaar. In één artikel worden logfiles geanalyseerd die zijn verzameld over een periode van 5 jaar. Niet alle onderzoeken vermelden het aantal leerlingen waarmee experimenten zijn uitgevoerd.

In de gevonden literatuur komen alle vormen van programma's die we in sectie 2 hebben beschreven voor. Er is een duidelijke voorkeur voor visueel programmeren, en dan weer met name, maar niet uitsluitend, voor het programmeren in Scratch. In de onderstaande tabellen geven we de aantallen artikelen waarin experimenten met de verschillende vormen van programmeren worden beschreven, en de aantallen artikelen waarin experimenten met verschillende soorten uitvoer worden beschreven. Het valt op dat sommige artikelen verschillende experimenten beschrijven die gebruik maken van verschillende vormen van programmeren en uitvoer.

Vorm	Aantal
Blokken	9
Visueel	37
Tekstueel	10
Unplugged	4

Uitvoer	Aantal
Robots	21
Grafisch	35
Tekstueel	2
Unplugged	2

In paragraaf 5.4 over de relatie tussen leeftijd en vormen van programmeren gaan we hier verder op in.

Het doel van de literatuurstudie is de volgende vragen te beantwoorden:

Wat weten we over de effecten van programmeeronderwijs op programmeervaardigheden van leerlingen tot 12 jaar? Deelvragen:

- Wat is bekend over de opbrengsten van dit programmeeronderwijs?
- Zijn opbrengsten verschillend voor verschillende leerlingen (jongens versus meisjes, voorkennis, sociaaleconomische status, interesse, leeftijd)?

Een deel van de gevonden studies beschrijven de resultaten aan de hand van niet vergelijkend kwantitatieve gegevens van één groep leerlingen die meestal een specifieke vorm van programmeeronderwijs gebruikt. Hoewel deze studies ook waardevolle resultaten laten zien voor de onderwijspraktijk, willen we hier voorzichtig met de interpretatie van de resultaten zijn. Doordat het vaak geen vergelijkend onderzoek betreft, kunnen andere factoren de resultaten mede bepalen. Ook worden soms de meningen van de onderzoekers of de docenten (aan de hand van hun observaties) gerapporteerd in de resultaten of is het aantal deelnemers laag en zijn de conclusies dus niet generaliseerbaar. Vanwege de methodologische kwaliteit van de studies en het verschil van de gehanteerde methoden moeten de resultaten van deze review voorzichtig geïnterpreteerd worden.

5.2 Verschil tussen jongens en meisjes

Zeven geselecteerde artikelen focussen specifiek op verschillen in leeropbrengst en/of motivatie van programmeeronderwijs bij jongens en meisjes (Baytak et al. [2011], Bruckman et al. [2002], Javidi and Sheybani [2009], Martinez et al. [2015], Merkouris and Chorianopoulos [2015], Mouza et al. [2016], Scharf et al. [2012]). De auteurs verwachten dat meisjes minder gemotiveerd en/of minder leren van programmeeronderwijs, omdat programmeren toch vaak gekarakteriseerd wordt als een activiteit die vooral door jongens wordt uitgevoerd en gewaardeerd.

Drie artikelen vinden dat er geen verschillen zijn tussen jongens en meisjes, zowel wat betreft leeropbrengsten (Scharf et al. [2012], Javidi and Sheybani [2009], Mouza et al. [2016]) als motivatie (Scharf et al. [2012]). Bruckman et al. [2002] analyseren de scripts gemaakt over een periode van 5 jaar tussen jongens en meisjes bij het gebruik van het tekstuele programma MOOSE ($n = 457$,

46% meisjes en 54% jongens). Jongens presteren beter dan meisjes. Regressieanalyses laten zien dat hogere scores van jongens volledig verklaard worden door de langere time-on-task en meer voorkennis bij het programmeeronderwijs. Andere artikelen laten zien dat meisjes na een programmeeronderwijsinterventie meer programmeerconcepten leren dan jongens (Martinez et al. [2015], Merkouris and Chorianopoulos [2015]). Merkouris and Chorianopoulos [2015] laten bijvoorbeeld zien dat meisjes beter presteren in de `als ... dan´ opdrachten dan jongens (18 jongens en 18 meisjes, gemiddeld 12 jaar). Dezelfde auteurs vonden in een vergelijkend experiment ook dat het programmeren van robots motiverender werkt dan het programmeren op een desktop computer. Met enige voorzichtigheid concluderen we dat er voor het programmeeronderwijs geen verschillen in motivatie tussen jongens en meisjes zijn, en dat meisjes soms meer leren van het programmeeronderwijs. Mouza et al. [2016] bestuderen het verschil tussen jongens en meisjes ($n = 52$, 30 jongens en 22 meisjes) bij het leren van belangrijke programmeerconcepten in Scratch gedurende 9 weken. Er zijn geen verschillen tussen jongens en meisjes, behalve bij vragen over *computer confidence* en *gender equity*. Jongens zijn meer *confident*, en meisjes scoren hoger op *gender equity* (meisjes kunnen net zo goed leren programmeren als jongens).

Vijf geselecteerde artikelen beschrijven experimenten met onderwijsinterventies waar alleen meisjes aan deelnemen (Kelleher et al. [2007], Aritajati et al. [2015], AlSulaiman and Horn [2015], Dimond et al. [2009], Marcu et al. [2010]). Het doel van deze experimenten is te onderzoeken of meisjes enthousiast gemaakt kunnen worden voor het leren programmeren, en om te verkennen of ze ook daadwerkelijk leren programmeren. In drie onderwijsinterventies wordt gebruik gemaakt van programmeeromgevingen of opdrachten die inspelen op interesses van meisjes, zoals sociale interactie (informeel maar ook via chat) en mode. Meisjes ervaren deze gender-geörienteerde aspecten van de tool als leuk, wat positief bijdraagt aan de motivatie van deze meisjes om de beginselen van het programmeren te leren. Kelleher et al. [2007] beschrijven een experiment waarin 88 meisjes visuele programma's construeren, en daarnaast werken met een variant waarin allerlei animaties (voor het vertellen van verhalen waarin virtuele characters figureren) makkelijk zijn te produceren. De resultaten laten geen verschil in programmeervaardigheden zien, maar wel een verschil in interesse in programmeren, waarbij de variant met animaties in hogere scores resulteert.

De twaalf artikelen laten zien dat, in tegenstelling tot een vaak geschetst beeld, zowel jongens als meisjes goed kunnen leren programmeren, en dat ze plezier beleven aan het leren programmeren.

5.3 Sociaaleconomische status

Programmeeronderwijs wordt vaak ingezet bij verbreding van het onderwijs voor plusklassen (Bell et al. [2008], Price and Barnes [2015]. Price and Barnes [2015]) onderzoeken het verschil tussen programmeren met blocks ($n = 19$)

en het programmeren met tekst ($n = 14$), in een plusklas met 12-jarige kinderen. Ze vinden geen significante verschillen in het leren programmeren, maar leerlingen die met blokken programmeren hebben minder tijd nodig om taken uit te voeren. Een aantal artikelen kijkt juist naar de vraag of programmeeronderwijs ook effectief kan zijn voor kinderen met een lage sociaaleconomische achtergrond (Javidi and Sheybani [2009], Marcu et al. [2010]). In vergelijking met de interventies in veel andere geselecteerde artikelen zijn de interventies beschreven in deze artikelen vrij intensief of langdurig, namelijk een minder intensief driejarig programma (Javidi and Sheybani [2009]) en een intensief zomerkamp van vier weken (Marcu et al. [2010]). De beschreven experimenten laten met behulp van pre- en posttests zien dat de programmeerinterventies een positief effect hebben op de interesse in en de leeropbrengsten van programmeren bij de doelgroep met een lagere sociaaleconomische status.

5.4 Leeftijd en vormen van programmeren

Zoals eerder beschreven onderscheiden we vier vormen van programmeren in de geselecteerde artikelen: unplugged, blokken, visueel en tekstueel. Daarnaast zijn er ook vier verschillende soorten uitvoer die door programma's worden gegenereerd: robots, grafisch, tekstueel, of unplugged. We bestuderen of het gebruik van verschillen programmeervormen gerelateerd is aan andere kenmerken van leerlingen, met name aan leeftijd.

Programma's gevormd door blokken en waarvan de uitvoer wordt getoond met behulp van robots worden enkel gebruikt bij relatief jongere doelgroepen (3–9 jaar, totaal $n = 181$) (Sullivan and Bers [2016], Scharf et al. [2008], Scharf et al. [2012], Wang et al. [2013], Wang et al. [2015], Wang et al. [2011], Wyeth and Purchase [2003], Wyeth [2008]). De gevonden artikelen beschrijven vooral kwalitatieve onderzoeken die laten zien dat jonge kinderen diverse programmeerconcepten snel leren door met behulp van blokken programma's te construeren die robots of andere elektronische objecten aansturen, en dat kinderen daar plezier aan beleven. Unplugged programmeermethoden worden ook gebruikt om andere vormen van programmeren te introduceren. Bijvoorbeeld Fessakis et al. [2013] bereiden leerlingen van 8–10 jaar voor op het schrijven van tekstuele programma's met behulp van unplugged programmeermethoden, zodat ze alvast kennismaken met de symbolen en logica van programmeren.

De meest voorkomende vorm van programmeren is visueel programmeren, waarbij de output van een programma grafisch is (26 artikelen, waarvan 15 Scratch gebruiken). Deze vormen worden gebruikt door doelgroepen van uiteenlopende leeftijden, waarbij het merendeel van de leerlingen rond de 8 jaar en ouder is, en dus wat ouder lijkt te zijn dan de leerlingen die gebruik maken van blokken of unplugged methoden.

We hebben relatief weinig studies gezien die zich richten op tekstueel leren programmeren in het basisonderwijs (Bruckman and De Bonte [1997], Bruck-

man et al. [2002], Dimond et al. [2009], Fessakis et al. [2013], Gregg et al. [2012], Serafini [2011], Boytchev [2011], Hsiao et al. [2011]). De leeftijd van de deelnemers aan de experimenten beschreven in deze artikelen ligt tussen de 8 en 15 jaar. Om tekstueel te kunnen programmeren moet een leerling kunnen lezen en schrijven, wat verklaart dat deze vorm van programmeren niet veel gebruikt wordt met jongere kinderen.

Okita [2014] vergelijkt twee groepen: een groep die eerst tekstueel en dan visueel leert programmeren, en een groep die eerst visueel en dan tekstueel leert programmeren. Eerst tekstueel en vervolgens visueel leren programmeren leidt tot betere resultaten bij 41 kinderen van 9–11 jaar. De auteur verklaart dit door te stellen dat tekstueel leren programmeren helpt om vaardigheden te ontwikkelen die makkelijker toegepast kunnen worden in andere situaties.

5.5 Vormen van ondersteuning bij het leren programmeren

Ondersteuning wordt vaak gepresenteerd als een belangrijke factor voor de motivatie van leerlingen. Kafai et al. [2012] concluderen dat het geven van feedback door bijvoorbeeld lesbegeleiders, medestudenten, experts en mensen uit de verbonden programmeercommunity aan leerlingen van 12 tot 14 jaar ($n = 14$) tijdens het programmeren in Scratch bijdraagt aan hun motivatie. Uit een experiment met 15 meisjes (10–15 jaar) blijkt, uit de observaties van de onderzoekers, dat het chatten over hun programmeeropdrachten bij het tekstueel programmeren goed is voor de motivatie, maar niet bijdraagt aan het leerproces (Dimond et al. [2009]). Bruckman and De Bonte [1997] onderzoeken het gebruik van Moose (een vorm van tekstueel programmeren) en concluderen ook dat gebruik kunnen maken van experts en medestudenten de sfeer in de klas ($n = 41$, 9–12 jaar) verbetert en een positief effect heeft op de voortgang. De auteurs concluderen ook dat samenwerkend leren (een vorm van interactie) bijdraagt aan het leren.

Drie geselecteerde artikelen laten zien dat leerlingen gebaat zijn bij ondersteunende instructies. Hsiao et al. [2011] vergelijken een groep 12-jarige leerlingen die ondersteunende instructies krijgt (bladen met extra vragen over het probleem) met een groep die deze niet krijgen ($n = 66$) bij het tekstueel programmeren gedurende 18 weken. De groep die instructies krijgt scoort significant beter en hebben een beter begrip van programmeren. Baytak and Land [2011] laten zien dat 11-jarige leerlingen ($n = 10$) geen boolean expressies in Scratch kunnen veranderen zonder hulp.

Harms et al. [2013] onderzoeken de effecten van tutorials bij het remixen (het veranderen, aanpassen of corrigeren van een programma dat door iemand anders is geschreven) bij 40 leerlingen (23 meisjes en 17 jongens, van 10 tot 16 jaar) die leren visueel programmeren. De experimentele groep krijgt een automatische gegenereerde tutorial die de deelnemer helpt bij het reconstrueren van het gemixte stuk van het programma. Leerlingen in de experimentele groep presteren significant beter in een transfer opdracht dan leerlingen in de

controle groep (F[2,37], $p < 0.05$). Er zijn echter geen verschillen tussen de groepen wat betreft interesse/ plezier (F[1,38] = 0.06, $p = 0.81$) of gepercipieerde competentie (F[1,38] = 0.617, $p = 0.44$).

Fessakis et al. [2013] laten met behulp van een kleinschalig onderzoek zien dat de helft van de leerlingen ($n = 10$, 5–6 jaar) geen ondersteuning nodig heeft bij het leren programmeren van een tekstueel programma. Drie leerlingen kunnen met behulp van ondersteuning de opdrachten voltooien, en twee leerlingen slagen er ondanks de hulp niet in om de opdrachten af te ronden. Voordat de leerlingen met het tekstueel programma aan de slag gaan, wordt de sessie met unplugged materialen geïntroduceerd, wat de resultaten zou kunnen verklaren. Doerschuk et al. [2009] voeren een explorerende studie uit met 26 'middle-school' leerlingen van een culturele minderheid gedurende één dag. Universitaire studenten die ook bij minderheidsgroepen behoren (bijvoorbeeld uit culturele minderheden) begeleiden de leerlingen tijdens de lessen. Zelfreport scores van computerkennis nam toe van 11.04 in de pretoets naar 19.36 in de posttoets ($t = 8.296$, $p < 0.00$). De auteurs concluderen dat de achtergrond van de begeleiders deze resultaten mede kunnen verklaren (de studenten zouden zich beter gerepresenteerd gevoeld hebben). Er worden geen significante verschillen gevonden in interesse, mogelijk vanwege de al hoge scores in de pretest.

Su et al. [2015] onderzoeken de effecten van het toevoegen van annotaties bij het leren programmeren met Scratch ($n = 36$, 12 jaar). De resultaten lieten zien dat het aantal annotaties en leerprestaties een positieve correlatie hebben (r=0.552, $p < 0.01$). Ook het aantal teruggekeken annotaties en leerprestaties laten een positieve correlatie zien ($r = 0.433$, $p < 0.01$). Tevens geven de leerlingen aan dat annotaties maken voor hun gevoel helpt, omdat ze meer nadruk leggen op wat ze niet snappen en daarop kunnen focussen tijdens de les.

Feng and Chen [2014] onderzoeken de effecten van gestructureerd versus *problematizing scaffolding* (hulp in de vorm van sturende vragen) met 232 leerlingen (11–12 jaar) die met Scratch werken. Dit experiment laat zien dat gestructureerd scaffolding leidt tot hoger programmeerbegrip met niet specifieke doelen dan met specifieke doelen (F(1,227) = 10.405, $p < 0.01$, $n^2 = 0.084$).

5.6 Andere didactische aanpakken

Gujberova and Kalas [2013] stellen op basis van een explorerende studie met 128 leerlingen (8–10 jaar) dat ontdekkend leren goed werkt om te leren programmeren. Leerlingen werken zelfstandige met unplugged programmeermaterialen in stapsgewijs oplopende moeilijkheid. Op basis van veelal kwalitatieve gegevens concluderen de auteurs dat het belangrijk is dat de opdrachten een oplopende moeilijkheidsgraad hebben als de studenten zelfstandig leren programmeren.

Aanpakken zoals project-gebaseerd leren blijken goed te werken. In een expe-

riment met Scratch met 107 leerlingen in groep 7 en 8 (61% meisjes, 39% jongens) (Sáez-López et al. [2016]) concluderen de auteurs dat project-gebaseerd leren de leerlingen motiveert. Leerlingen scoorden hoog op *perceived usefulness* ($M = 4.5$ op een schaal van 1 tot 5). Su et al. [2014] beschrijven een vergelijkend onderzoek met 135 leerlingen van gemiddeld 12 jaar tussen een traditionele aanpak en probleem-gebaseerd leren. Zij concluderen dat leerlingen die met een probleem-gebaseerd leren-aanpak werken beter scoren op programmeerbegrip in termen van syntax (grammatica) en semantiek (betekenis van de code) dan leerlingen die met een traditionele aanpak werken.

Drie artikelen exploreren de effecten van het maken van een game. Scharf et al. [2012] concluderen in een kleinschalig onderzoek dat bij het maken van een spel met blokken er geen verschil tussen jongens ($n = 10$) en meisjes ($n = 10$, 6–9 jaar) in termen van plezier en programmeerbegrip is. Javidi and Sheybani [2009] concluderen in een experiment met 89 leerlingen (51 jongens en 38 meisjes) dat het ontwerpen van een spel met LEGO een effectieve aanpak kan zijn voor leerlingen met een lagere sociaaleconomische status. In een kleinschalig kwalitatief onderzoek met 10 leerlingen (6 jongens en 4 meisjes, groep 7) die gedurende 21 dagen met Scratch werken (Baytak et al. [2011]) komt naar voren dat veel leerlingen bij het maken van een spel dezelfde strategie gebruiken. Deze strategie houdt in dat leerlingen *template* (voorbeeld) programma's verkennen en gebruiken bij het maken van hun eigen spel. Ze gebruiken een *trial-and-error* strategie om hun programma vervolgens de verbeteren. Een nadeel van deze aanpak kan zijn dat het spel dat een leerling maakt sterk bepaald wordt door de beschikbare voorbeelden.

6 Conclusies

Steeds meer scholen onderwijzen programmeren, maar veel scholen vinden het moeilijk om programmeren een duidelijk plek in het curriculum te geven. Op basis van een literatuurstudie, waarbij 48 artikelen uiteindelijk geselecteerd zijn, hebben we gekeken naar hoe leerlingen in het Primair onderwijs (PO) (en specifieker, tussen 6 en 12 jaar) leren programmeren. De analyse van de geselecteerde artikelen had als doel de volgende vragen te beantwoorden:

Wat weten we nu over de effecten van programmeeronderwijs op programmeervaardigheden van leerlingen tot 12 jaar? Deelvragen waren:

- Wat is bekend over de opbrengsten van dit programmeeronderwijs?
- Zijn opbrengsten verschillend voor verschillende leerlingen (jongens versus meisjes, voorkennis, sociaaleconomische status, interesse, leeftijd)?

Ondanks het volgen van de selectiecriteria en het dubbelchecken van twijfelgevallen komen we tot de conclusies dat, vanwege de methodologische kwaliteit van de studies en het verschil van de gehanteerde methoden, de resultaten van deze review voorzichtig geïnterpreteerd moeten worden (zie paragraaf 5.1 voor meer toelichting). Er is meer vergelijkend onderzoek naar de

effecten op programmeervaardigheden nodig om hardere conclusies te kunnen trekken. Ondanks deze beperkingen worden wel een aantal patronen geïdentificeerd die docenten, onderzoekers, onderwijskundigen en beleidsmakers verder kunnen ondersteunen in hun zoektocht naar manieren om programmeeronderwijs vorm te geven.

Er bestaan verschillende definities van het begrip programmeren. In sectie 2 wordt het begrijp programmeren ingekaderd en worden vier vormen van programma's die kinderen in het PO kunnen construeren toegelicht: tekstueel, visueel, unplugged, of (fysieke) blokken.

De artikelen vinden in het algemeen geen verschillen tussen jongens en meisjes, zowel wat betreft leeropbrengsten als motivatie. Resultaten laten voorzichtig zien dat meisjes even goed als jongens kunnen leren programmeren en tegen dezelfde problemen aanlopen. Bij een analyse van de tekstuele MOOSE programma's gemaakt over een periode van 5 jaar, waarbij programma's van jongens en meisjes werden vergeleken, werd wel gevonden dat jongens betere programma's kunnen schrijven. Aan de andere kant lieten nadere analyses zien dat deze verschillen verklaard worden door de langere *time-on-task* en meer voorkennis bij het programmeeronderwijs van jongens. Bij een studie met 30 jongens en 22 meisjes bleek dat jongens hoger op *computer confidence* en meisjes hoger op *gender equity* scoorden. Wat betreft het leren programmeren bij leerlingen met lagere socialeconomische status laten de beschreven experimenten voorzichtig zien dat programmeerinterventies een positief effect hebben op de interesse in en de leeropbrengsten van programmeren bij deze doelgroep.

Programma's gevormd door blokken en waarvan de uitvoer wordt getoond met behulp van robots worden gebruikt bij relatief jongere doelgroepen (leeftijd 3-9 jaar). Visuele programma's met veelal grafische output worden het meest gebruikt in het PO. Deze vorm wordt gebruikt door doelgroepen van uiteenlopende leeftijden, waarbij het merendeel van de leerlingen rond de 8 jaar en ouder is, en dus wat ouder lijkt te zijn dan de leerlingen die gebruik maken van blokken of unplugged methoden. We hebben slechts een beperkt aantal artikelen gezien die zich richten op het leren tekstueel programmeren in het basisonderwijs. De leeftijd van de deelnemers aan de experimenten ligt tussen de 8 en 15 jaar. Unplugged programmeermethoden worden het minst gebruikt in de geselecteerde artikelen. De leeftijd ligt daar tussen de 5 en 6 jaar, of hoger, maar wordt dan gebruikt om tekstuele programma's te introduceren.

Een vergelijkende studie liet zien dat eerst tekstueel programmeren en vervolgens visueel tot betere resultaten leidt dan eerst visueel en dan tekstueel programmeren. Ondersteuning in de vorm van feedback (van docenten, experts of medestudenten) draagt bij aan de motivatie. Ondersteuning in de vorm van tutorials en advies helpt leerlingen bij het leren programmeren, vooral bij 'moeilijke taken' (zoals geavanceerde *logic blocks* (Wyeth [2008])). Het is nog onduidelijk hoeveel abstractie kinderen in de basisschoolleeftijd aankunnen. Andere factoren die goede effecten op het leren programmeren kunnen heb-

ben zijn bijvoorbeeld samenwerkend leren en interactie (*face to face* of *online*, bijvoorbeeld via *chat*) en het maken en terugkijken van eigen annotaties. Annotaties en prestaties correleren, ook teruggekeken annotaties en leren. Een mogelijke verklaring is dat als leerlingen eigen annotaties terugkijken ze het schrijven van een programma beter begrijpen.

Algemene conclusies

Deze literatuurstudie laat resultaten en patronen zien die kunnen bijdragen aan het vormgeven van programmeeronderwijs in het curriculum van het Primair onderwijs.

Hoewel we 48 artikelen gevonden hebben waarin wordt onderzocht hoe leerlingen in de leeftijd van 6 tot 12 jaar leren programmeren, is er weinig kwantitatief vergelijkend onderzoek uitgevoerd naar de leeropbrengsten van verschillende programmeeronderwijsinterventies. Dit maakt het lastig om sterk onderbouwde conclusies te trekken op basis van de gereviewde artikelen.

Er zijn veel programmeermaterialen beschikbaar voor kinderen, zelfs in situaties waarin geen computer beschikbaar is. Hoewel programmeren dus ook zonder computers onderwezen kan worden, voegen elektronische apparaten (scherm, robot, geluid, of licht) wel aspecten toe die veel leerlingen leuk vinden, en maken ze directe feedback mogelijk.

Het is belangrijk een leerling te ondersteunen bij het construeren van een programma. De rol van de docent is dus erg belangrijk! Daarbij komt onmiddellijk de vraag naar boven hoe goed docenten voorbereid zijn op het ondersteunen van een leerling bij het construeren van programma's. Docentprofessionalisering is nodig om leerlingen te helpen. We denken dat er voldoende laagdrempelige tools aanwezig zijn om ook docenten te professionaliseren op het gebied van programmeren. Maar er moet nog wel gewerkt worden aan een omgeving die docenten die leren programmeren om later programmeeronderwijs te gaan verzorgen te ondersteunen.

Als leerlingen programmeren zijn ze niet alleen maar gebruikers van ICT, maar dragen ze zelf ook bij. Daarmee hopen we dat ICT verandert van iets magisch uit een machine naar technologie die je kan gebruiken, veranderen, of zelfs zelf construeren.

Referenties

Sarah AlSulaiman and Michael S. Horn. Peter the fashionista?: Computer programming games and gender oriented cultural forms. In *Proceedings CHI PLAY '15: the 2015 Annual Symposium on Computer-Human Interaction in Play*, pages 185–195. ACM, 2015.

Chulakorn Aritajati, Mary Beth Rosson, Joslenne Pena, Dana Cinque, and Ana Segura. A socio-cognitive analysis of summer camp outcomes and expe-

riences. In *Proceedings SIGCSE '15: the 46th ACM Technical Symposium on Computer Science Education*, pages 581–586. ACM, 2015.

Ahmet Baytak and Susan M. Land. An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom. *Educational Technology Research and Development*, 59(6):765–782, 2011.

Ahmet Baytak, Susan M. Land, and Brian K. Smith. Children as educational computer game designers: An exploratory study. *Turkish Online Journal of Educational Technology - TOJET*, 10(4):84–92, 2011.

Scott Bell, Linda Heeler, and Phillip Heeler. A preliminary report on the use of robots with elementary school students. *J. Comput. Sci. Coll.*, 23(4):263–268, 2008.

Pavel Boytchev. Wild programming – one unintended experiment with inquiry based learning. In Ivan Kalaš and Roland T. Mittermeir, editors, *Informatics in Schools. Contributing to 21st Century Education: 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011, Bratislava, Slovakia, October 26-29, 2011. Proceedings*, pages 1–8. Springer, 2011.

Neil Christopher Charles Brown, Michael Kölling, Tom Crick, Simon Peyton Jones, Simon Humphreys, and Sue Sentance. Bringing computer science back into schools: lessons from the UK. In *Proceedings SIGCSE '13: the 44th ACM technical symposium on Computer science education*, pages 269–274, 2016.

Amy Bruckman and Austina De Bonte. Moose goes to school: A comparison of three classrooms using a cscl environment. In *Proceedings CSCL '97: the 2Nd International Conference on Computer Support for Collaborative Learning*, pages 20–27. International Society of the Learning Sciences, 1997.

Amy Bruckman, Carlos Jensen, and Austina DeBonte. Gender and programming achievement in a cscl environment. In *Proceedings CSCL '02: the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community*, pages 119–127. International Society of the Learning Sciences, 2002.

Jill P. Dimond, Sarita Yardi, and Mark Guzdial. Mediating programming through chat for the olpc. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, pages 4465–4470. ACM, 2009.

Peggy Doerschuk, Jiangjiang Liu, and Judith Mann. Inspired computing academies for middle school students: Lessons learned. In *Proceedings TAPIA '09: the Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations*, pages 52–57. ACM, 2009.

Chia-Yen Feng and Ming-Puu Chen. The effects of goal specificity and scaffolding on programming performance and self-regulation in game design. *British Journal of Educational Technology*, 45(2):285–302, 2014.

G. Fessakis, E. Gouli, and E. Mavroudi. Problem solving by 56 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63:87–97, 2013.

Chris Gregg, Luther Tychonievich, James Cohoon, and Kim Hazelwood. Ecosim: A language and experience teaching parallel programming in elementary school. In *Proceedings SIGCSE '12: the 43rd ACM Technical Symposium on Computer Science Education*, pages 51–56. ACM, 2012.

Shuchi Grover and Roy Pea. Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1):38–43, 2013.

Monika Gujberova and Ivan Kalas. Designing productive gradations of tasks in primary programming education. In *Proceedings WiPSE '13: the 8th Workshop in Primary and Secondary Computing Education*, pages 108–117. ACM, 2013.

Kyle J. Harms, Dennis Cosgrove, Shannon Gray, and Caitlin Kelleher. Automatically generating tutorials to enable middle school children to learn programming independently. In *Proceedings IDC '13: the 12th International Conference on Interaction Design and Children*, pages 11–19. ACM, 2013.

Sheng-Che Hsiao, Janet Mei-Chuen Lin, and Jiin-Cherng Kang. Learning to program in kpl through guided collaboration. *US-China Education Review*, 8 (1):89–97, 2011.

Giti Javidi and Ehsan Sheybani. Digispired: Digital inspiration for interactive game design and programming. *J. Comput. Sci. Coll.*, 24(3):144–150, 2009.

Yasmin B. Kafai, Quinn Burke, and Chad Mote. What makes competitions fun to participate?: The role of audience for middle school game designers. In *Proceedings IDC '12: the 11th International Conference on Interaction Design and Children*, pages 284–287. ACM, 2012.

Caitlin Kelleher, Randy Pausch, and Sara Kiesler. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings CHI '07: the SIGCHI Conference on Human Factors in Computing Systems*, pages 1455–1464. ACM, 2007.

Sze Yee Lye and Joyce Hwee Ling Koh. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41:51–61, 2014.

Gabriela Marcu, Samuel J. Kaufman, Jaihee Kate Lee, Rebecca W. Black, Paul Dourish, Gillian R. Hayes, and Debra J. Richardson. Design and evaluation of a computer science and engineering course for middle school girls. In *Proceedings SIGCSE '10: the 41st ACM Technical Symposium on Computer Science Education*, pages 234–238. ACM, 2010.

Cecilia Martinez, Marcos J. Gomez, and Luciana Benotti. A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In *Proceedings ITiCSE '15: the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 159–164. ACM, 2015.

Alexandros Merkouris and Konstantinos Chorianopoulos. Introducing computer programming to children through robotic and wearable devices. In *Proceedings WiPSCE '15: the Workshop in Primary and Secondary Computing Education*, pages 69–72. ACM, 2015.

Jesús Moreno-León and Gregorio Robles. Code to learn with Scratch? A systematic literature review. In *Proceedings EDUCON '16: the IEEE Global Engineering Education Conference*, pages 150–156, 2016.

Chrystalla Mouza, Alison Marzocchi, Yi-Cheng Pan, and Lori Pollock. Development, implementation, and outcomes of an equitable computer science after-school program: Findings from middle-school students. *Journal of Research on Technology in Education*, 48(2):84–104, 2016.

Sandra Y. Okita. The relative merits of transparency: Investigating situations that support the use of robotics in developing student learning adaptability across virtual and physical computing platforms. *British Journal of Educational Technology*, 45(5):844–862, 2014.

Thomas W. Price and Tiffany Barnes. Comparing textual and block interfaces in a novice programming environment. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 91–99. ACM, 2015.

José-Manuel Sáez-López, Marcos Román-González, and Esteban Vázquez-Cano. Visual programming languages integrated across the curriculum in elementary school: A two year case study using scratch in five schools. *Computers & Education*, 97:129–141, 2016.

Florian Scharf, Thomas Winkler, and Michael Herczeg. Tangicons: Algorithmic reasoning in a collaborative game for children in kindergarten and first class. In *Proceedings IDC '08: the 7th International Conference on Interaction Design and Children*, pages 242–249. ACM, 2008.

Florian Scharf, Thomas Winkler, Claudia Hahn, Christian Wolters, and Michael Herczeg. Tangicons 3.0: An educational non-competitive collaborative game. In *Proceedings IDC '12: the 11th International Conference on Interaction Design and Children*, pages 144–151. ACM, 2012.

Giovanni Serafini. Teaching programming at primary schools: Visions, experiences, and long-term research prospects. In Ivan Kalaš and Roland T. Mittermeir, editors, *Informatics in Schools. Contributing to 21st Century Education: 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011, Bratislava, Slovakia, October 26-29, 2011. Proceedings*, pages 143–154. Springer, 2011.

Addison Y. S. Su, Stephen J. H. Yang, Wu-Yuin Hwang, Chester S. J. Huang, and Ming-Yu Tern. Investigating the role of computer-supported annotation in problem-solving-based teaching: An empirical study of a scratch programming pedagogy. *British Journal of Educational Technology*, 45(4):647–665, 2014.

Addison Y. S. Su, Chester S. J. Huang, Stephen J. H. Yang, T. J. Ding, and Y. Z. Hsieh. Effects of annotations and homework on learning achievement: An empirical study of scratch programming pedagogy. *Journal of Educational Technology & Society*, 18(4):331–343, 2015.

Amanda Sullivan and Marina Umaschi Bers. Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in prekindergarten through second grade. *International Journal of Technology and Design Education*, 26(1):3–20, 2016.

Danli Wang, Cheng Zhang, and Hongan Wang. T-maze: A tangible programming tool for children. In *Proceedings IDC '11: the 10th International Conference on Interaction Design and Children*, pages 127–135. ACM, 2011.

Danli Wang, Yunfeng Qi, Yang Zhang, and Tingting Wang. Tanpro-kit: A tangible programming tool for children. In *Proceedings IDC '13: the 12th International Conference on Interaction Design and Children*, pages 344–347. ACM, 2013.

Danli Wang, Lan Zhang, Yunfeng Qi, and Fang Sun. A tui-based programming tool for children. In *Proceedings ITiCSE '15: the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 219–224. ACM, 2015.

Peta Wyeth. How young children learn to program with sensor, action, and logic blocks. *Journal of the Learning Sciences*, 17(4):517–550, 2008.

Peta Wyeth and Helen C. Purchase. Using developmental theories to inform the design of technology for children. In *Proceedings IDC '03: the 2003 Conference on Interaction Design and Children*, pages 93–100. ACM, 2003.

Halil Yurdugül and Petek Aşkar. Learning programming, problem solving and gender: A longitudinal study. *Procedia - Social and Behavioral Sciences*, 83:605–610, 2013.