EFFECTEN VAN PROGRAMMEERONDERWIJS OP COMPUTATIONAL THINKING

REVIEWSTUDIE

Joke Voogt | Saskia Brand-Gruwel | Johan van Strien

Windesheim

Universiteit van Amsterdam

Open Universiteit
welten-instituut.ou.nl

EFFECTEN VAN PROGRAMMEERONDERWIJS OP COMPUTATIONAL THINKING

REVIEWSTUDIE

Joke Voogt, Saskia Brand-Gruwel** en Johan van Strien***

Reviewstudie in opdracht van het Nationaal Regieorgaan Onderwijsonderzoek (NRO)

NRO NATIONAAL REGIEORGAAN ONDERWIJSONDERZOEK

* Hogeschool Windesheim, Zwolle / Universiteit van Amsterdam
**Open Universiteit, Welten-instituut, onderzoekscentrum voor leren doceren en Technologie, Heerlen

Inhoudsopgave

1 Inleiding

In 2006 riep Janet Wing (2006) op om in het primair en voortgezet onderwijs aandacht te besteden aan computational thinking skills. Hoewel er nog geen breed gedragen algemene definitie van computational thinking bestaat, worden computational thinking skills beschouwd als een set van probleemoplossende vaardigheden en houdingen waarbij men gebruikmaakt van concepten uit de informatica (Lye & Koh, 2014) en die betrekking hebben op het denken in stappen, het ordenen van informatie, het besef van volgordelijkheden en het modelleren van gegevens (Fisser & Strijker, 2016). Het kunnen abstraheren en analytisch denken wordt daarbij gezien als essentieel kenmerk van computational thinking (Wing, 2008). De gedachte om al in het primair en voortgezet onderwijs aandacht aan computational thinking te besteden is dat informatica heeft gezorgd voor grote vooruitgang in de wetenschap en het dagelijks leven, en dat het niet meer voldoende is om lerenden pas in het hoger onderwijs te laten kennismaken met aspecten van computational thinking (Barr & Stephenson, 2011). Wing's oproep heeft geleid tot hernieuwde aandacht in diverse landen voor het inpassen van vaardigheden die te maken hebben met informatie- en communicatietechnologie in het curriculum. Zo is in Engeland in 2014 'Computing' ingevoerd in het basiscurriculum, en is in Australië 'Digital Technologies' een verplicht onderdeel van het curriculum. In beide curricula zijn computational thinking skills (CT) een belangrijk onderdeel. Ook in de Nederlandse context werd Wing's oproep opgepakt. Er verscheen in 2013 een rapport van de KNAW over het belang van het vak informatica in het voortgezet onderwijs, waarin ook verwezen wordt naar CT en in het adviesrapport van het platform Onderwijs2032 wordt CT expliciet genoemd als onderdeel van digitale geletterdheid, wat een verplicht kernonderdeel in het hernieuwde curriculum zou moeten worden.

Bij computational thinking gaat het om een vaardigheid en manier van denken die in de informatica gangbaar is om problemen aan te pakken, daarom wordt vaak de link met programmeren gelegd. Programmeren wordt gezien als een fundamentele vaardigheid voor informatici en als een belangrijk middel om te werken aan taken die computational skills vereisen (Grover & Pea, 2013). Maar in feite gaan computational thinking skills om meer dan programmeren, en is probleemoplossend denken een vaardigheid die in meerdere vakgebieden belangrijk is (Barr & Stephenson, 2011). Aspecten van CT kunnen dan ook heel goed in andere vakgebieden aan de orde komen, zoals het stelonderwijs en de natuurwetenschappen (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013; Voogt, Fisser, Good, Mishra, & Yadav, 2015; Wolz, Stone, Pearson, Pulimood, Switzer, 2011). Dat CT breder is dan alleen programmeren wordt algemeen erkend (Wing, 2008), en tegelijkertijd is ook duidelijk dat aspecten van programmeren een bijdrage kunnen leveren aan de ontwikkeling van CT-skills, maar dat dit niet zonder meer gebeurt (Voogt et al., 2015).

Er is ook nog veel discussie over hoe CT zich verhoudt tot andere 21st-century skills, zoals information literacy, waarbij bijvoorbeeld het kritisch denken en ordenen van informatie van belang is (e.g., Brand-Gruwel & Stadtler, 2011). Zo wordt ook samenwerken gezien als een belangrijk element in CT (The College Board, 2012), maar wordt dit ook gezien als een 21st-century skill op zichzelf. Met name in de VS zijn er diverse initiatieven om CT verder te operationaliseren, zodat curriculumontwikkeling mogelijk is. Daarbij is het van belang te kijken naar instrumentarium voor het inzichtelijk maken van de CT-skills van leerlingen en studenten. Zie bijvoorbeeld de initiatieven van de Computer Science Teaching Association (https://csta.acm.org/Curriculum/sub/CompThinking.html). Zo heeft SLO in haar uitwerking van een leerlijn voor CT zich voornamelijk gebaseerd op deze bronnen.

Er is echter nog weinig onderzoek naar de effecten van programmeeronderwijs op CT in het primair en voortgezet onderwijs, zoals blijkt uit de reviewstudie van Lye en Koh (2014). Lye en Koh hanteren een indeling van CT in concepts, practices en perspectives. Bij computational concepts gaat het om het aanleren van kennis over concepten die horen bij CT, zoals 'variabelen' en 'loops'. Computational practices heeft betrekking op hoe lerenden probleemoplossend werken tijdens het programmeren zelf; het betreft de toepassing van CT-skills, denken in stappen, het kunnen ordenen enzovoort. Bij computational perspectives, tot slot, gaat het erom hoe lerenden aankijken tegen de relatie met henzelf, anderen en de technologische wereld in relatie tot programmeren. Hierbij komt het houdingsaspect van CT naar voren. Vragen die daarbij gesteld kunnen worden zijn of je zelf vindt dat de invloed van technologie groot is op het dagelijks leven, of je daar zelf een steentje aan kunt bijdragen en of je vindt dat leren programmeren erbij kan helpen. Lye en Koh vonden slechts negen studies in het primair en voortgezet onderwijs die aan hun criteria voldeden, waarvan er slechts twee gingen over computational practices en twee over computational perspectives. De meeste studies gingen over het aanleren van computational concepts.

Om het onderwijs meer inzicht te geven in de effecten van programmeeronderwijs op CT van leerlingen is deze reviewstudie uitgevoerd. De resultaten kunnen onderwijsontwikkelaars en docenten helpen bij het maken van keuze over de wijze waarop programmeeronderwijs kan worden ontworpen om CT van leerlingen te bevorderen. In deze reviewstudie worden drie onderzoeksvragen beantwoord.

Onderzoeksvragen

1 Welke evidentie is er dat programmeeronderwijs bijdraagt aan de ontwikkeling van computational thinking skills bij kinderen tussen 4 en 18 jaar (primair en voortgezet onderwijs)?
2 Onder welke condities kan deze ontwikkeling het beste worden gerealiseerd?
3 Welk instrumentarium is er beschikbaar om de ontwikkeling ten aanzien van computational thinking skills in kaart te brengen?

Deze studie bouwt verder op een eerdere NRO-reviewstudie van Jeuring, Corbalan, Van Es, Van Leeuwestein en Van Montfort (2016), maar hier ligt de focus niet alleen op het primair onderwijs maar ook op het voortgezet onderwijs, en het gegeven dat niet alleen wordt gekeken naar de effecten op programmeren, maar op een breder palet van computational thinking skills.

2 Methode

Om de onderzoeksvragen te kunnen beantwoorden, is een systematische literatuurstudie naar gepeer-reviewde artikelen uitgevoerd binnen de wetenschappelijke databases Web of Science, Scopus en ERIC. Bij het zoeken zijn drie sets met termen gebruikt: één rondom 'computational thinking', één rondom ' programmeren' en één rondom de doelgroep 'K-12' (zie Appendix I voor de exacte zoektermen).

Het zoeken leverde 126 hits op in Web of Science, 10 in ERIC en 88 in Scopus. Gefilterd op het voorkomen van dubbele resultaten, leverde dit 196 unieke hits op. Bij het bepalen of artikelen al dan niet zouden worden meegenomen in de analyse, werden de volgende inclusiecriteria geformuleerd:

- Het betreft empirisch onderzoek (experimenteel of casestudie; geen literatuurreviews of theoretische artikelen).
- De uitkomstvariabelen hebben betrekking op computational thinking skills, en niet alleen op affectieve maten.
- De studies betreffen interventies of onderwijs dat gericht is op programmeren.

Twee van de onderzoekers namen onafhankelijk van elkaar de gevonden artikelen door en kwamen in gezamenlijk overleg uit op 25 geselecteerde artikelen, waarvan er 23 beschikbaar waren als full-text. Uit de referentielijsten van deze 23 artikelen werden nog vijf aanvullende bronnen geselecteerd die niet bij het zoekproces naar voren waren gekomen. In totaal zijn er dus 28 studies meegenomen in de review.

De geselecteerde studies zijn door paren van twee onderzoekers gelezen. Hierbij werden de doelgroep, het aantal deelnemers, de context, het onderzoeksdesign, de gebruikte programmeertaal, de interventie, de uitkomstvariabelen evenals de operationalisatie daarvan en de gevonden effecten genoteerd in een bronnentabel. De uitkomsten werden vergeleken en waar nodig aangevuld of bijgesteld.

3 Resultaten

In dit hoofdstuk geven we antwoord op de drie onderzoeksvragen. We lopen de drie vragen door en geven aan wat de verschillende studies ten aanzien van de drie vragen opleveren.

3.1 Evidentie dat programmeeronderwijs bijdraagt aan CT

De eerste onderzoeksvraag luidde: Welke evidentie is er dat programmeeronderwijs bijdraagt aan de ontwikkeling van computational thinking skills (kennis vaardigheden en attitudes) bij kinderen tussen 4 en18 jaar (primair en voortgezet onderwijs)? In de uitwerking maken we onderscheid in verschillende operationalisaties van het concept CT. We kijken naar studies over effecten van programmeeronderwijs op generieke kennis en vaardigheden zoals probleemoplosvaardigheden (3.1.1), naar studies over effecten op kennis en vaardigheden in computer science (3.1.2), naar vakinhoudelijke kennis en vaardigheden (3.1.3) en naar affectieve aspecten (attitudes) (3.1.4).

3.1.1 Computational thinking skills: generieke vaardigheden

In veertien studies in de dataset is nagegaan of er sprake is van een effect van programmeeronderwijs op generieke vaardigheden, te weten probleemoplosvaardigheden (9 studies), creatieve vaardigheden (1 studie), analytisch redeneren (1 studie), kwantitatief redeneren (1 studie), kritisch denken (1 studie), reflectieve vaardigheden (1 studie) en metacognitieve taken (1 studie). Een overzicht van de studies wordt gepresenteerd in tabel 1.

Tabel 1 Effecten van programmeeronderwijs op generieke vaardigheden: overzicht van de studies.

Auteur	Onderwijssector[1]	Uitkomstvariabele	Effect[2]
Fessakis et al	kleuters	probleemoplosvaardigheden	positief effect
Clements & Gullo	onderbouw po	creatief denken	significant positief effect
		reflectieve vaardigheid	significant positief effect
		metacognitieve taak	significant positief effect
		cognitieve ontwikkeling	geen effect
Klahr & Carver	bovenbouw po	probleemoplosvaardigheden	positief effect
Pardamean et al.	bovenbouw po	probleemoplosvaardigheden	significant positief effect
Kalelioglu & Gulbahar	bovenbouw po	probleemoplosvaardigheden	geen effect
Kalelioglu	bovenbouw po	probleemoplosvaardigheden	geen effect
Linn	onderbouw vo	probleemoplosvaardigheden	geen effect
Yang & Chang	onderbouw vo	kritisch denken	significant positief effect (ook op retentietest)
Ke	onderbouw vo	analytisch en kwantitatief redeneren	positief effect
Akcaoglu	onderbouw vo	probleemoplosvaardigheden: - systeemanalyse & ontwerp	positief effect
		- besluitvorming	positief effect
		- troubleshooting	geen effect
Akcaoglu & Koehler	onderbouw vo	probleemoplosvaardigheden: - systeemanalyse & ontwerp	significant positief effect
		- besluitvorming	significant positief effect
		- debugging	significant positief effect
Atmatzidou & Demetriadis	onderbouw vo/ bovenbouw vo	probleemoplosvaardigheden (generalisatie)	positief effect voor leerlingen bovenbouw vo
Ennis	bovenbouw vo	probleemoplosvaardigheden	geen effect
Palumbo & Reed	bovenbouw vo	probleemoplosvaardigheden	significant positief effect

1 Onderwijssector is een indicatie van de leeftijd van de leerlingen in de studie en gerelateerd aan de corresponderende onderwijssector in Nederland.
2 In de overzichtstabellen wordt alleen van een significant positief effect gesproken als er een vergelijking is met een controlegroep

Probleemoplosvaardigheden

Primair onderwijs

Zes studies naar het effect van probleemoplosvaardigheden vonden plaats in het primair onderwijs (inclusief het kleuteronderwijs). Twee daarvan zijn uitgevoerd vóór 1995. Vier studies laten een positief effect zien van programmeeronderwijs op probleemoplosvaardigheden. In twee studies werd geen verschil gevonden. Fessakis et al. (2013) deden onderzoek naar het effect van programmeeronderwijs op probleemoplosvaardigheden bij kleuters. Zij vonden dat de helft van de tien kleuters de problemen (programmeren van de weg naar het doel) kon oplossen zonder een enkele hint en drie kleuters dit konden na één hint. Klahr & Carver (1988) onderzochten wat het effect is van gerichte instructie in strategieën gericht op debuggen van een Logoprogramma op een probleemoplostaak. Uit hun bevindingen blijkt dat leerlingen die deze instructie hadden gekregen de probleemoplostaak beter uitvoerden dan leerlingen die de instructie niet hadden gekregen, en dit gold ook voor een taak die zonder logo moest worden opgelost. Deze taak is weergegeven in box 1.

Box 1 Probleemoplostaak (ontleend aan Klahr & Carver, 1988)

Twee verhuizers helpen mevrouw Fisser bij het verhuizen naar een nieuwe woning. Zij vroeg de verhuizers om haar meubels neer te zetten en ze gaf de verhuizers een lijst met aanwijzingen. De verhuizers volgden de aanwijzingen precies op, maar toch waren de meubels niet neergezet zoals mevrouw Fisser dat wenste.
Je krijgt twee plaatjes en de lijst met aanwijzingen. In het eerste plaatje zie je hoe mevrouw Fisser wilde dat de verhuizers haar meubels zouden neerzetten. In het tweede plaatje zie je hoe de verhuizers de meubels hebben neergezet.
Zoek op welke fouten mevrouw Fisser heeft gemaakt in de lijst met aanwijzingen die ze aan de verhuizers gaf.

Pardamean, Supraryanto & Evelyn (2015) vonden een significant positief verschil tussen leerlingen die Logo-instructie hadden gekregen op generieke probleemoplosvaardigheden, in vergelijking met leerlingen die geen Logo-instructie hadden gekregen. De gebruikte test meet twee aspecten van generieke probleemvaardigheid: ruimtelijk inzicht en logisch redeneren. Het effect is groter op ruimtelijk inzicht, dan op logisch redeneren. Kaleiloglu & Gulbahar (2014) en Kaleiloglu (2015) rapporteren over onderzoek naar het effect van programmeeronderwijs, met respectievelijk Scratch en de code.org-website. Zij vonden geen verschil in tussen voor- en nameting in generieke probleemoplosvaardigheden.

Voortgezet onderwijs

Acht studies zijn uitgevoerd in het vo. Drie daarvan zijn uitgevoerd vóór 1995. Twee van de acht studies vonden geen effect van programmeeronderwijs op probleemoplosvaardigheden. De andere vijf studies vonden positieve effecten.
Linn (1985) beschrijft een studie naar de effecten van het programmeren in Basic op leerlingen uit de onderbouw van het voortgezet onderwijs. Zij vond geen effect van de interventie op probleemoplosvaardigheden. Akcaoglu (2014) en Akcaoglu en Koehler (2014) onderzochten probleemoplosvaardigheden bij leerlingen uit de onderbouw vo die digitale spellen leerden ontwerpen. Zij maakten onderscheid tussen verschillende probleemtypen: systeemanalyse & ontwerp, besluitvorming en troubleshooting. In beide onderzoeken werd een significant positief effect op systeemanalyse & ontwerp en besluitvorming gevonden. Akcaoglu en Koehler (2014) vonden wel een effect op troubleshooting, maar in het onderzoek van Akcaoglu (2014) werd dit effect niet gevonden. Als verklaring voor deze laatste uitkomst wijzen de onderzoekers op het pre-experimentele design van de studie van Akcaoglu (2014). Atmatzidou & Demitriadis (2016) deden onderzoek naar het effect van onderwijs in robotics bij leerlingen in de onder- en bovenbouw van het vo. Probleemoplosvaardigheden werden geoperationaliseerd als generaliseren van het probleemoplosproces naar een brede set van uiteen-

lopende problemen. In een toets die halverwege en aan het eind van de interventie werd afgenomen bleek een toename in probleemoplosvaardigheden van leerlingen uit de bovenbouw van het voortgezet onderwijs. Dit was niet het geval bij leerlingen uit de onderbouw van het voortgezet onderwijs, maar hun score was bij aanvang al hoger dan die van de leerlingen uit de bovenbouw. De onderzoekers verklaarden dit door de groepssamenstelling. De leerlingen uit de bovenbouw van het vo waren voornamelijk jongens.

Palumbo & Reed (1991) en Ennis (1994) deden onderzoek naar de effecten van programmeeronderwijs op probleemoplosvaardigheden in de bovenbouw van het vo. Palumbo & Reed (1991) vonden een significant positief effect op probleemoplosvaardigheid tussen leerlingen die programmeeronderwijs hadden genoten en leerlingen die een onderwijs in digitale basisvaardigheden hadden gevolgd. Ennis (1994) onderzocht of het aanleren van een algemene probleemoplosstrategie in combinatie met leren programmeren in Basic meer bijdraagt aan generieke probleemoplosvaardigheden dan het alleen leren programmeren. Zij vond geen significant verschil op een test die probleemoplosvaardigheden meet. Wel vond Ennis dat leerlingen de aangeleerde strategie handig vonden voor het schrijven van programma's en dat hun programma's een creatievere oplossing boden voor het programmeerprobleem.

Andere generieke vaardigheden

Clements en Gullo (1984) deden onderzoek naar het effect van programmeeronderwijs op het creatieve denken en de reflectieve vaardigheden van leerlingen uit de onderbouw van het primair onderwijs. Zij vergeleken een groep die programmeeronderwijs volgde met een groep die computerondersteunde instructie kreeg. Zij vonden een significant verschil op creatief denken, reflectieve vaardigheden, en metacognitieve taken voor de groep die programmeeronderwijs volgde, maar geen verschil in cognitieve ontwikkeling tussen beide groepen. Yang & Chang (2013) onderzochten het effect van game making binnen het vak biologie op kritisch denken bij leerlingen uit de onderbouw van het voortgezet onderwijs. Zij vonden een significant verschil tussen de experimentele en controle conditie, ook op de retentietoets. Ke (2014) heeft kwalitatief onderzoek uitgevoerd naar het effect van het ontwerpen van games in de context van het wiskundeonderwijs. Zij observeerde ontwerpteams van leerlingen uit de onderbouw van het vo en analyseerde de ontwerpproducten. Uit haar onderzoek blijkt dat de interventie bijdraagt aan analytisch denken (in 25 procent van de gecodeerde gebeurtenissen lieten de leerlingen dit zien) en dat leerlingen daarbij gebruik maken van kwantitatief redeneren (het modelleren van een authentiek probleem met wiskundige symbolen en expressies) (in 15.5 procent van de gecodeerde gebeurtenissen lieten de leerlingen dit zien).

3.1.2 Computational thinking skills: computer science-concepten en -vaardigheden

Vijf studies deden onderzoek naar effecten van programmeeronderwijs op computational thinking skills, die expliciet werden gerelateerd aan computer science (CS)-concepten en -vaardigheden. Hoewel er in dit onderzoek vaak wel specifieke CS-concepten en –vaardigheden bij leerlingen worden ontwikkeld, werden de effecten vastgesteld op het geheel van de computational thinking skills die werden ontwikkeld, en niet op de onderscheiden concepten en vaardigheden. Tabel 2 geeft een overzicht.

Tabel 2 Effecten van programmeeronderwijs op CS-concepten en - vaardigheden: overzicht van de studies.

Auteur	Onderwijssector[1]	Uitkomstvariabelen	Effect[2]
Jenson & Droumeva	bovenbouw PO	CS-concepten	positief effect
Saez et al.	bovenbouw PO	CS-concepten en computational practices	significant positief effect
Moreno-Leon et al.	bovenbouw PO /onderbouw VO	computational practices	positief effect
Linn	onderbouw VO	computational practices	effect varieert
Liu et al.	bovenbouw VO	computational practices	studie 1: positief effect: studie 2: positief effect

1 Onderwijssector is een indicatie van de leeftijd van de leerlingen in de studie en gerelateerd aan de corresponderende onderwijssector in Nederland
2 In de overzichtstabellen wordt alleen van een significant positief effect gesproken als er een vergelijking is met een controlegroep

Primair onderwijs

Drie studies vonden plaats in de bovenbouw po. Jenson en Droumeva (2016) beschrijven een pilotstudie naar effecten van onderwijs in game making bij leerlingen. Zij rapporteerden een positief effect tussen voor- en nameting op een test die basiskennis CS-concepten meet. Saez et al. (2016) deden onderzoek naar programmeren met Scratch in de bovenbouw van het po. Zij vonden een positief verschil tussen voor- en nameting en op een toets die computer science-concepten (sequenties, herhaalde uitvoering, conditionele uitvoering, gelijktijdige uitvoering, coördinatie, event handling, keyboard input), en computational practices (experimenteren en iteraties) meet. Eveneens werd een significant verschil gevonden op de nameting tussen experimentele - en controlegroep. Moreno-Leon et al. (2015) lieten leerlingen uit de bovenbouw van po en de onderbouw van vo een eigen programma testen en verbeteren op basis van feedback gegenereerd door het programma (Dr. Scratch). In de test van het programma wordt gekeken hoe in het programma gebruik wordt gemaakt van de volgende CS-concepten: abstractie, probleemontleding, logisch denken, synchronisatie, parallelle uitvoering, algoritmes voor control flow, interactiviteit en datarepresentatie. Uit dit onderzoek bleek dat de programma's verbeterden na de feedback. Het effect was het grootst voor leerlingen met een gemiddeld beginniveau en voor leerlingen uit de onderbouw van het vo.

Voortgezet onderwijs

In de onderbouw van het vo vonden twee studies plaats. Over de resultaten van het onderzoek van Moreno-Leon et al. (2015) is hierboven al gerapporteerd. Linn (1985) deed in de jaren tachtig onderzoek naar het effect van programmeren. Uit haar onderzoek bleek een positief effect op het leren kennen van de programmeertaal (Basic), maar het effect op het ontwerpen van een programma in Basic varieerde. In klassen waar expliciete instructie werd gegeven in het ontwerpen van programma's in Basic werd een positief effect gemeten, dit in tegenstelling tot klassen waar wel onderwijs in Basic was, maar geen expliciete instructie.
Liu et al. (2013) deden onderzoek in de bovenbouw van het vo. Zij vonden een significant positief verschil tussen pre- en posttest op basisconcepten uit de computer science. In een vervolgstudie werden klassen vergeleken die een virtuele robot programmeerden met klassen die met het zelfde curriculum een fysieke robot programmeerden. Er was geen verschil tussen de condities: beide groepen scoorden significant hoger op de nameting in vergelijking met de voormeting. Wel hadden de klassen met de fysieke robot aanzienlijk meer tijd nodig om het curriculum te voltooien.

Specifieke CS-concepten en -vaardigheden

In acht studies werden de effecten van programmeeronderwijs op specifieke CS-concepten en -vaardigheden gemeten. Een overzicht wordt gegeven in tabel 3.

Tabel 3 Effecten van programmeeronderwijs op specifieke CS-concepten en - vaardigheden: overzicht van de studies

Concept	Auteurs	Onderwijssector[1]	Effect[2]
Complexe systemen	Berland & Wilensky	onderbouw vo	positief effect
Debugging	Bers et al.	Kleuters	positief effect
	Klahr & Carver	bovenbouw po	positief effect
Correspondence & sequentie	Bers et al.	kleuters	positief effect
	Kazakoff et al	kleuters	significant positief effect
Control flow	Bers (control flow)	kleuters	positief effect
	Meerbaum et al. (diverse constructen)	onderbouw vo	effect varieert voor deelvaardigheden
	Grover et al. (algoritme)	onderbouw vo	positief effect
	Liu et al. (algoritme)	bovenbouw vo	geen effect (studie 1); positief effect (studie 2)
	Atmatzidou & Demetriadis (algoritme)	onderbouw vo/ bovenbouw vo	positief effect voor leerlingen bovenbouw vo
Abstraheren	Atmatzidou Demetriadis	onderbouw vo/ bovenbouw vo	geen effect
Moduleren	Atmatzidou Demetriadis	onderbouw vo/ bovenbouw vo	positief effect voor leerlingen onderbouw en bovenbouw vo
Decompositie	Atmatzidou Demetriadis	onderbouw vo/ bovenbouw vo	positief effect voor leerlingen onderbouw en bovenbouw vo

1 Onderwijssector is een indicatie van de leeftijd van de leerlingen in de studie en gerelateerd aan de corresponderende onderwijssector in Nederland
2 In de overzichtstabellen wordt alleen van een significant positief effect gesproken als er een vergelijking is met een controlegroep

Berland en Wilensky (2015) gingen na wat het effect is van een interventie waarin leerlingen uit de onderbouw van het vo leerden om robots te programmeren. Een klas programmeerde fysieke robots, terwijl de andere klas met virtuele robots programmeerde. De onderzoekers vonden een positief effect voor beide condities op hun begrip van complexe systemen, maar op een andere manier. De leerlingen die fysieke robots programmeerden begrepen deze systemen vanuit de robot die zij programmeerden. Zij wilden het gedrag van hun robot optimaliseren, wat resulteerde in complexere robots. De leerlingen die virtuele robots programmeerden keken naar het totale systeem. Zij maakte meer, maar simpeler circuits.
Twee studies onderzochten het effect van programmeeronderwijs op de vaardigheid van leerlingen om fouten in een computerprogramma te identificeren en op te lossen (debuggen). Klahr & Carver (1988) deden dat in de bovenbouw van het po. Zij gaven leerlingen gerichte instructie in strategieën gericht op debuggen van een Logoprogramma en vonden een significant positief effect tussen voor- en nameting op de vaardigheid om te debuggen. Bers et al. (2013) leerden kleuters om een robot te programmeren en stelden het effect op debuggen vast. Zij concludeerden dat kleuters deze vaardigheid meer dan gemiddeld beheersten.

In het onderzoek van Bers et al (2014) werd ook nagegaan of kleuters de vaardigheid om instructies te kiezen (correspondence) en om deze in de juiste volgorde te plaatsen beheersen (sequentie) om een robot te programmeren. Zij vonden dat ongeveer 75 procent van de kleuters deze vaardigheid beheersten als het gaat om eenvoudige acties. Kazakoff et al. (2013) deden onderzoek naar dezelfde interventie, het programmeren van een robot. Zij vroegen kleuters om plaatjes in de juiste volgorde te leggen om zo een verhaal te vertellen. Kleuters die de interventie hadden gevolgd verbeterden zichzelf significant in vergelijking met een controlegroep. In het onderzoek van Bers et al. (2014) werd ook gekeken naar de vaardigheid om de volgorde waarin de robot instructies uitvoert, te beheersen (control flow). In dit onderzoek gaat het dan met name om het meerdere keren uitvoeren van dezelfde handeling (loop) en de uitvoering onder bepaalde voorwaarden ('if' en 'if not'). Uit het onderzoek bleek dat ongeveer 59 procent van de kleuters deze vaardigheid na de lessen beheerste. Er werd geen verschil gevonden tussen de vaardigheid om loops te gebruiken en de vaardigheid om condities ('if') toe te passen. Wel hadden meer kleuters problemen met het toepassen van de negatieve conditie ('if not').

Meerbaum et al. (2013) deden onderzoek in de onderbouw van het vo naar de ontwikkeling van bepaalde CS-concepten en -vaardigheden in een curriculum waarin Scratch wordt gebruikt. Zij onderzochten diverse CS-concepten en -vaardigheden die onder control flow (de (niet-lineaire) uitvoering van de instructies in een computerprogramma) kunnen worden gerangschikt, te weten: initialiseren, herhaalde uitvoering, voorwaardelijke uitvoering, event handling, variabelen, communicatie via berichten en gelijktijdige uitvoering (concurrency). Er werd gekeken naar effecten op een interimtoets en een eindtoets. De onderzoeksresultaten toonden aan dat de meeste leerlingen in staat waren om een redelijk niveau van begrip van CS-concepten te realiseren. Moeilijkheden werden aangetroffen in het onderwijzen van specifieke vaardigheden, met name vaardigheden die meer abstractie vereisten, zoals herhaalde uitvoering, variabelen en gelijktijdige uitvoering.

Algoritme (een set regels om een probleem op te lossen) is een vergelijkbaar concept als control flow. Drie studies deden onderzoek naar het effect van programmeeronderwijs op dit CS-concept. Grover en collega's (2015) gebruikten het concept algoritme in een interventie waarin leerlingen uit de onderbouw van het vo met Scratch programmeren. Zij operationaliseerden algoritme als: sequentie, herhaaldelijke uitvoering, conditionele uitvoering. Zij vonden een significant positief effect van de interventie op het kunnen toepassen van algoritmen. De leerlingen hadden het meeste moeite met het toepassen van herhaalde uitvoering, gevolgd door het toepassen van voorwaardelijke uitvoering (zogenaamde 'als-dan'-redeneringen). Liu et al. (2013) deden een interventie waarin leerlingen uit de bovenbouw van het vo fysieke dan wel virtuele robots leerden programmeren. In een eerste studie vonden zij een positief effect voor algoritmisch denken in die klas die zowel virtuele als ook fysieke robots had geprogrammeerd. In een vervolgstudie werd een positief effect gevonden voor beide condities In een tweede studie werden klassen vergeleken die een virtuele robot programmeerden met klassen die met het zelfde curriculum een fysieke robot programmeerden. Er werd nu een significant positief effect gevonden op algoritmisch denken voor beide condities. In het onderzoek van Atmatzidou en Demetriadis (2016) naar het effect van onderwijs in robotics bij leerlingen in de onder- en bovenbouw van het VO werden diverse CS-concepten onderzocht. Naast generaliseren (zie probleemoplosvaardigheden) werd gekeken naar algoritmen (programmeerinstructies), abstraheren (analyse), moduleren (herhaald gebruik van procedures) en decompositie (opsplitsen van het probleem). Zij vonden een significant positief effect tussen interim- en eindtoets voor alle leerlingen op moduleren en decompositie en voor bovenbouwleerlingen op algoritmen. Atmatzidou en Demetriadis (2016) vonden geen effect op abstraheren.

3.1.3 Effecten op andere vakinhoudelijke kennis en vaardigheden

Vier studies onderzochten het effect van programmeeronderwijs op vakinhoudelijke kennis en vaardigheden. Een samenvatting van deze studies wordt gepresenteerd in tabel 4.

Tabel 4 Effecten van programmeeronderwijs op vakinhoudelijke kennis en vaardigheden: overzicht van de studies.

Auteur	Onderwijssector - vak	Uitkomstvariabele	Effect
Yang & Chang	onderbouw vo - biologie	kennis	significant positief effect (ook op retentietest)
Fessakis et al	Kleuters	rekenvaardigheden	positief effect
Ke	onderbouw vo - wiskunde	wiskundig denken; houding t.o.v. wiskunde	positief effect
Saez et al.	bovenbouw po – kunstgeschiedenis	kennis	positief effect

Fessakis et al. (2013) deden onderzoek naar kleuters. Uit de observaties van de lessen en het interview met de leerkracht bleek dat het programmeeronderwijs ook bijdroeg aan de rekenvaardigheden van de leerlingen, zoals getalbegrip, tellen, richting, hoek. Saez et al. (2016) deden onderzoek naar programmeren met Scratch in het kader van kunstgeschiedenisonderwijs in de bovenbouw van het po. Uit hun onderzoek bleek dat de leerlingen door de interventie inzicht en begrip (in de terminologie van Bloom) van kunstgeschiedenis hadden verworven.

Twee studies, beiden in de onderbouw van het voortgezet onderwijs, beschrijven effecten van het ontwerpen en programmeren van spellen voor respectievelijk biologische (Yang & Chang, 2013) en wiskundige vakinhouden (Ke, 2014). Yang & Chang (2013) vonden een significant positief effect op een kennistoets. Ke (2014) onderzocht de attitude ten opzichte van wiskunde en vond een significant effect op houding (zelfvertrouwen, waarde, plezier en motivatie) (paarsgewijze t-test $t(62) = 2{,}56$, $p = 0{,}01$). Uit de kwalitatieve data bleek ook een positief effect op het herkennen en reflecteren van wiskunde in de samenleving en op het wiskundig denken. Meer dan de helft van de leerlingen geeft aan over wiskunde te hebben geleerd door de interventie.

3.1.4 Effecten op affectieve variabelen

Attitude is bij kleuters alleen zijdelings meegenomen in de studie van Fessakis et al. (2013). Het betrof een casestudie waarin werd geconcludeerd dat de kinderen gemotiveerd waren, omdat ze na anderhalf uur werken aan activiteiten aangaven door te willen gaan met nieuwe activiteiten.

Bij de studies in de onderbouw van het primair onderwijs is attitude niet meegenomen. Bij twee studies uitgevoerd in de bovenbouw van het primair onderwijs is attitude gemeten. In de studie van Jenson en Droumeva (2016) bleek het doorlopen van een programmeerinstructie gedurende anderhalve week het zelfvertrouwen ten aanzien van programmeren temperde en dat dit vooral bleek bij meisjes. Maar interessant is wel dat zowel jongens als meisjes de ervaring met het programmeren zien als 'fun' en 'exciting'. Ze geven aan dat het uitdaging was iets te doen wat ze nog nooit hadden gedaan en tegelijkertijd zeggen ze dat het ook een moeilijke vaardigheid is.

De studie van Sáez-López et al. (2016) meet attitude met een schaal van vijf vragen. De score op deze schaal was hoog en er kon worden geconcludeerd dat studenten enthousiast en gemotiveerd waren om te werken volgens de aangeboden systematiek.

In de studie van Ke (2014), uitgevoerd in de bovenbouw, is nagegaan wat het effect van de interventie (maken van een game) is op de attitude van leerlingen ten aanzien van wiskunde. Daarvoor is de 'Attitudes towards Math Inventory' (ATMI, Tapia & Marsh, 2004) gebruikt, bestaande uit veertig items met een vijfpuntsschaal. De resultaten laten zien dat het ontwerpen van een game een positieve invloed heeft op de attitude van leerlingen ten aanzien van wiskunde.

In de studie van Pellas en Peroutseas (2016) is onderzocht hoe leerlingen in de onderbouw van het voortgezet onderwijs het ervaren om te werken in een game-gebaseerde omgeving. Uit het onderzoek bleek de aandacht en de betrokkenheid van de leerlingen tijdens het werken in de omgeving hoog. Ze vonden het ook aantrekkelijk om in een dergelijke omgeving te werken.

In de studie van Kalelioglu en Gulbahar (2014) zijn kinderen uit de bovenbouw van het primair onderwijs geïnterviewd nadat ze programmeeronderwijs hadden gevolgd. Op de vraag of ze het leuk vonden om te programmeren was het antwoord van alle leerlingen 'ja'. Ook waren ze zeker in voor het verder leren van deze vaardigheid.

3.2 Condities waaronder programmeeronderwijs bijdraagt aan CT

De tweede onderzoekvraag luidde: Onder welke condities kan deze ontwikkeling het best kunnen worden gerealiseerd? Voor het ontwerpen van onderwijs is het van belang te weten onder welke condities programmeeronderwijs bijdraagt aan de ontwikkeling van CT. Een overzicht van de kenmerken van de interventies die hebben geleid tot de positieve effecten is te vinden in Appendix II. Appendix II is geordend naar onderwijsniveau.
Verreweg de meeste interventies vonden plaats in de bovenbouw van het primair onderwijs en de onderbouw van het voortgezet onderwijs. We bespreken daarom de kenmerken van interventies die een positief effect sorteerden in deze twee onderwijsniveaus. Daarbij moeten we bedenken dat niet alle artikelen de interventie in detail beschrijft.

Bovenbouw primair onderwijs

In de interventies die effectief bleken, is gebruikt gemaakt van Logo (2x), Scratch (2x), Dr. Scratch (1x) en Game maker studio (1x). De duur van de interventies varieert sterk: van 9 uur Jenson & Droumeva, 2016) tot ruim 33 uur (Klahr & Carver, 1988). Over de opbouw en inhoud van de interventies kan het volgende worden gezegd. Er wordt gestart met het leren kennen van de programmeeromgeving (inclusief de syntax), veelal door expliciete instructie. Daarna worden begrippen en procedures geïntroduceerd. Dit gebeurt in oplopende complexiteit. Met name het concept procedure en het samenstellen van diverse procedures gericht op de oplossing van een probleem wordt behandeld en geoefend. Jenson en Droumeva (2013) besteden daarnaast aandacht aan condities. Klahr en Carver (1988) aan een strategie om fouten in programma's op te sporen en op te lossen. Na elke introductie oefenen de leerlingen met de nieuwe (en al geleerde) begrippen en procedures. Leerlingen werken soms aan opdrachten geformuleerd door de docent of onderzoeker (Klahr & Carver, 1988; Pardamean et al., 2015) en soms aan zelf geformuleerde opdrachten (Jenson & Droumeva, 2016; Klahr & Carver, 1988). In de studie van Pardamean et al. (2015) wordt elke les afgesloten met een toets over het geleerde, in de studie van Klahr en Carver krijgen de leerlingen drie keer gedurende het traject een toets.

In alle studies werken de leerlingen samen aan de opdrachten in groepjes van twee. In de studie van Klahr en Carver (1988) en Pardamean et al. (2015 heeft de docent zowel een instructierol (introduceren van nieuwe concepten) als een begeleidende rol (tijdens het werken aan de opdrachten). In de studie van Jenson en Droumeva (2016) wordt de rol van de docent door de onderzoekers vervuld. Zij hebben vooral een instructierol. De leerlingen worden uitgedaagd zelf, of met de helpfunctie in de software, oplossingen te vinden voor problemen die zij tegenkomen (Jenson & Droumeva, 2016; Klahr & Carver, 1988).

Onderbouw voortgezet onderwijs

Verreweg de meeste interventies vonden plaats in de onderbouw van het voortgezet onderwijs. In deze interventies werd gebruikt gemaakt van Scratch (3x), Kodu (2x), Basic (1x), VBOT (1x), Digital Game Authorship (1x) en Lego mindstorms (1x). De duur van de interventies varieert aanzienlijk. Het is lastig om de precieze tijdsinvestering aan te geven, omdat deze met verschillende eenheden wordt aangeduid. De kortste interventie is beschreven in de studie van Ke (2014). Deze interventie beslaat zes sessies van elk één uur. De meest omvangrijke interventie is beschreven in de studie van Grover et al. (2015): 28 sessies van elk 55 minuten.

De opbouw en inhoud van de lessen in de diverse studies verschillen. In de studies van Ke (2014) en Yang en Chang (2013) wordt het programmeeronderwijs gekoppeld aan een vak (respectievelijk wiskunde en biologie) en maken de leerlingen een spel gelieerd aan dat vak. Naast het leren van vakinhoudelijke concepten leren leerlingen een spel ontwerpen, het ontwerp implementeren en toetsen. In het onderzoek van Yang en Chang (2013) maken ze daarbij ook kennis met aspecten van game design. Ook in de onderzoeken van Akcaoglu (2014) en Akcaoglu en Koehler (2014) wordt in de les systematisch aandacht besteed aan het ontwerpen van games. Daarnaast wordt tijd gereserveerd voor het leren identificeren en oplossen van fouten in een programma.

In twee studies leren de leerlingen een robot programmeren (virtueel dan wel fysiek). In de studie van Atmatzidou en Demetrios (2016) komen leerlingen op systematische wijze in aanraking met een aantal computer science-concepten: algoritme, sequentie, loop, control flow - (samengestelde) procedures. De opdrachten worden in de loop van de lessen steeds complexer. Dit laatste is ook het geval in het onderzoek van Berland en Wilensky (2015).
In het onderzoek van Meerbaum et al. (2013) en Grover et al. (2015) wordt in de lessen nadruk gelegd op het leren begrijpen en gebruiken van computer science-concepten, met name control flow. Bij Meerbaum et al. (2013) gaat het bijvoorbeeld om seriële, voorwaardelijke en gelijktijdige uitvoering. De leerlingen leren eerst een concept kennen, toepassen en ontwerpen, en pas daarna wordt een nieuw concept geïntroduceerd. Bij het onderzoek van Grover et al. (2015) wordt de te leren kennis en vaardigheden die in de les aan de orde komt ingebed in technologische ontwikkelingen die leerlingen in de maatschappij tegenkomen.

In de meeste onderzoeken werken leerlingen aan opdrachten die geformuleerd zijn door de docent (of de onderzoekers in hun rol als docent). Deze opdrachten hebben tot doel om met de geleerde concepten te oefenen. In sommige studies krijgen de leerlingen (daarnaast) ruimte om met de geleerde kennis aan eigen

opdrachten te werken (Akcaoglu, 2014); Akcaoglu & Koehler, 2014; Grover et al. (2015); Ke, 2014; Yang & Chang, 2013). In twee studies werken de leerlingen in grote teams van minimaal 5 leerlingen samen aan één spel (Ke, 2014; Yang & Chang, 2013). Soms werken leerlingen alleen (Berland & Wilensky, 2015) of kunnen ze kiezen (Grover et al., 2015). Van de andere studies is niet altijd bekend wat de groepsgrootte is.

De betrokken docenten en onderzoekers verzorgen expliciete instructie over de aan te leren concepten en procedures. De studie van Ke (2014) is hierop een uitzondering. In dit onderzoek hebben de docenten alleen een rol als begeleider. Daarnaast begeleiden ze de leerlingen als ze aan het werk zijn. In een aantal onderzoeken verzorgt de onderzoeker of een onderzoeksassistent de docentrol (Atmatzidou & Demetriadis, 2016; Grover et al., 2015) en soms hebben onderzoeksassistenten een begeleidende rol (Ke, 2014).

3.3 Meten van computational thinking skills

De derde onderzoeksvraag luidde: Welk instrumentarium is er beschikbaar om de ontwikkeling ten aanzien van computational thinking skills in kaart te brengen?
Voor het onderwijs is het van belang om te meten wat effecten zijn van het ontworpen onderwijs en dus is het van belang inzicht te hebben in de wijze waarop we CT kunnen meten. In deze sectie zullen we bespreken hoe in de verschillende onderwijsniveaus in de studies de vaardigheden zijn gemeten en bespreken we meetinstrumenten die bruikbaar zijn in de onderwijspraktijk.

Kleuters
De drie studies uitgevoerd bij kleuters laten elk een andere wijze zien van het bepalen van de mate waarin de kinderen kennis en vaardigheden hebben opgedaan.
De studie van Kazakoff et al. (2013) keek alleen naar het effect van programmeeronderwijs op sequentiëren oftewel logisch ordenen; dat aspect is gemeten met een toets waarin kinderen plaatjes moesten ordenen. Ze kregen vijftien verhaaltjes, steeds bestaande uit vier plaatjes waarbij het eerste plaatje is gegeven en de andere drie plaatjes in logische volgorde moeten worden gelegd, zodat het verhaal klopte (Baron et al, 1986).

Bij de studie van Fessakis et al. voerden de kinderen verschillende activiteiten uit en daarbij praatten de leerlingen samen over de uitgevoerde activiteiten. De video-opnames zijn uitgeschreven en geanalyseerd om na te kunnen gaan wat de leereffecten waren.
In de studie van Bers et al. (2014) werden vier concepten gemeten (debugging, correspondence between action and instructions, sequencing en control flow). Tijdens de lessen kwamen deze vier aspecten in meer of mindere mate voor en na elke les werd door een beoordelaar op een zespuntsschaal aangegeven in hoeverre concepten werden begrepen en activiteiten goed werden uitgevoerd (bijvoorbeeld van goede uitvoering tot geen poging).

Onderbouw primair onderwijs
De studie van Clements en Gullo (1984) is uitgevoerd in de onderbouw van het primair onderwijs. In deze studie is gekeken naar effecten van de interventie op generieke vaardigheden, zoals creatief denken en reflectievaardigheden. Voor deze generieke vaardigheden zijn hoofdzakelijk gestandaardiseerde testen gebruikt, zoals de Torrance Test of Creative Thinking (Torrance, 1972). Geen specifieke instrumenten voor computational thinking zijn meegenomen. Ook Kalelioglu (2015) keek in zijn studie naar algemene probleemoplossingsvaardigheden en deze werden gemeten met de Reflective Thinking Skill Scale Towards Problem Solving van Kizilkaya en Askar (2009). Beide studies gebruikten bestaande instrumenten voor het meten van algemene vaardigheden.

Bovenbouw primair onderwijs
Er zijn negen studies uitgevoerd in de bovenbouw van het primair onderwijs. De studies verschillen in de mate van gedetailleerdheid in de beschrijving van de gehanteerde meetinstrumenten.

Ennis (1995) richtte zich in het onderzoek op algemene vaardigheden die werden gemeten met de Cognitive Abilities Test (CAT): Non-Verbal Battery to measure problem-solving ability.

Kalelioglu en Gulbahar (2014) hebben onderzocht wat het effect was van een programmeerinterventie met Scratch op de probleemoplossingsvaardigheden van leerlingen uit de bovenbouw van het primair onderwijs. Ze hebben voor het meten van deze generieke vaardigheid gebruik gemaakt van de Problem Solving Inventory van Serin, Bulut Serin, en Saygili (2010). Daarnaast werden er focusgroepen gehouden.

In de studie van Jenson en Droumeva (2016) maakten de leerlingen een eigen game en werd gekeken naar effecten op media literacy en op attitude. Media literacy werd heel breed gemeten met een vragenlijst. Er zaten bijvoorbeeld items bij die gingen over gebruik van social media. Attitude werd gemeten met vijf vragen, zoals 'ik heb er vertrouwen in dat ik de taak kan afronden', 'ik voel me prettig' enzovoort. De leerlingen scoorden dat op een vijfpuntsschaal. Over deze schalen worden geen psychometrische gegevens gerapporteerd, waardoor niet te beoordelen is hoe betrouwbaar en valide de vragenlijsten zijn.

Ook de studie van Pardamean et al. (2015) was gericht op de generieke vaardigheid probleemoplossen. Deze werd gemeten met twee bestaande tests: Figural Problem Solving Test en de Logical Word Test. Daarnaast werd programmeren met LOGO beoordeeld door leerlingen individueel in een sessie van veertig minuten drie open problemen te laten oplossen. Gekeken werd in hoeverre leerlingen gebruik maakten van eenvoudige commando's, repeterende commando's, en procedures voor creëren van complexe commando's. Daarnaast werden geometrische concepten gemeten, zoals: rotatie, hoek van de rotatie, cirkel, rechthoek, driehoek. Leerlingen konden op de test maximaal 100 punten scoren. Er worden geen psychometrische gegevens gerapporteerd over dit meetinstrument.
De studie van Werner et al. (2014) werd uitgevoerd bij leerlingen van 10 tot 14 jaar en was gericht op het programmeren van games en het effect op CT. Daarbij werden de gemaakte games geanalyseerd. Game patterns werden onderzocht, waarbij men kan denken aan gebruik van verschillende constructen als gebruik van movements, sounds, dialog box en timers.

De focus in de studie van Klahr en Carver (1988) lag op de debugging-vaardigheid van leerlingen. Leerlingen kregen verschillende testen met steeds zes bugs die ze moesten oplossen. Vijf bugs waren semantisch van aard, wat wil zeggen dat het programma wel wordt uitgevoerd, maar de uitkomst niet goed is. Eén bug was syntactisch van aard, en betrof een spellingfout, een interpunctiefout of een spatiefout. Het programma kan dan ook niet worden gedraaid. Tijdens het oplossen van de tests dachten de leerlingen hardop en verwoordden hun denkproces. Deze 'hardopdenkprotocollen' werden samen met de opgenomen activiteiten op het scherm geanalyseerd om het proces van debuggen te ontrafelen.

In de studie van Moreno et al. (2015) werd een Dr. Scratch-workshop met een tijdsduur van één uur geëvalueerd aan de hand van de volgende vragen. De meeste schalen werden gemeten op een driepuntsschaal. Tabel 5 heeft de gestelde vragen weer.

Tabel 5 Vragen gesteld in de studie van Moreno et al. (2015).

Activity	Question
1 Visit the Dr. Scratch website:	a What do you think about the website? Do you find it attractive? b After reading the information on the website, what do you think Dr. Scratch can be used for?
2 Analyze one of your Scratch projects with Dr. Scratch.	a Is it easy to analyze projects with Dr. Scratch? b What was your score? c According to Dr. Scratch, what is the CT level for that score? d How did you feel when you saw the results? e Why?
3 From the results page, after analyzing a project, click on some of the links to receive information that could help you improve your code.	a Write the title of the page you clicked on. b Do you understand the information in the results page? c After reading the information, do you feel like trying something new?
4 Using the information that appeared in the help page you selected, try to improve your project by adding something new.	a Are the ideas and tips in the results page enough to improve your program? b After performing some modifications, analyze again your project with Dr. Scratch. What is the new score?
	a Do you have any other comments?

'Computational concepts and practices' werden onderzocht in de studie van Sáez-López et al. (2016). Deze twee aspecten werden gemeten met de Visual Blocks Creative Computing Test. Deze test bestaat uit veertig items. De studenten beantwoorden vragen met betrekking tot 'sequences, loops', 'conditional statements', 'parallel execution', 'coordination', 'event handling' en 'keyboard input' om 'computational concepts' te meten. Voor het beoordelen van computational practices werden vragen met betrekking tot 'experimenten and iterating' gesteld. Een meer gedetailleerde beschrijving van het instrument ontbrak. De psychometrische waarden kunnen als goed worden bestempeld. In deze studie werden ook de leerprocessen en de attitudes van de leerlingen gemeten. Dit is gedaan met een vragenlijst met vijf schalen en in totaal 24 items. Daarbij is gebruik gemaakt van een vijfpuntsschaal. In tabel 6 worden de vragen weergegeven.

Tabel 6 Vragenlijst gehanteerd in de studie van Sáez-López et al. (2016)

1 Active learning	• Learned many factual materials • Improved ability to communicate clearly • Became more interested in the subject • Participated actively • Assignments aided the student's learning
2 Contents in art history	• Understood artistic elements in paintings • Learned biographical and historical contents of Spanish painters • Increased cultural and artistic competence to understand paintings • Improved the ability to understand artistic expressions from different eras • Analyzed historical and artistic content in paintings
3 Computational concepts	• Understood sequences with combined characters, backgrounds, and elements • Included loops in programming to allow a proper multimedia product • Added parallelism and events that allow the creation of interface • Improved ability to share and play with the content created • Acquired the ability to communicate and express through the content created • Increased fun to learn art history with games and animations
4 Perceived usefulness	• The courseware increased the efficiency of my learning process • The courseware helped improve my learning performance • The courseware was useful
5 Enjoyment	• I was happy • I enjoyed the activity • I was enthusiastic • I felt motivated • I was relaxed and comfortable

De studie uitgevoerd door Zhong, Wang, Chen en Li (2016) had betrekking op hoe CT is te meten. Vanuit een theoretisch perspectief maakten ze onderscheid in drie soorten taken (open, semi-open en gesloten) en die drie soorten taken combineerden ze met het 'direction'-principe (forward en reverse). Dat leverde zes type taken op die zijn te onderscheiden: tabel 7 geeft een overzicht van de zes taken.

Tabel 7 Zes type taken gebruikt in de studie van Zhong et al. (2016).

1	The closed forward task, which is an unfinished task with a defined outcome and a defined process solely;
2	The semi-open forward task, which is an unfinished task with a defined outcome solely and an undefined or open process;
3	The closed reverse task, which is a troubleshooting task with a defined out- come and a defined process solely;
4	The semi-open reverse task, which is a troubleshooting task with a defined outcome solely and an undefined or open process;
5	The open task with a creative design report, which is a creative task with an open outcome and open process; and
6	The open task without a creative design report, which is a creative task with an open outcome and open process.

Deze zes taken zijn vervolgens uitgewerkt en gekoppeld aan de CT-vaardigheden die werden gemeten. Tabel 8 laat zien hoe de uitwerking is gemaakt. De taken zijn beschreven, de CT-vaardigheid is benoemd, evenals de wijze van beoordelen. Bij de beoordelingen werd een scoringsrubric gebruikt en werden op de verschillende aspecten scores gegeven.

Tabel 8 Uitwerking van de zes taken in de studie van Zhong et al. (2016).

Types of tasks	Goal or outcome	CT	Assessment
Task 1: The closed forward task	Task: Make the small rabbit eats off a green cauliflower. Specification: The small rabbit has the motion of stooping to eat, and after which a cauliflower will be disappeared, then the small rabbit will recover to stand erectly.	Sequences, testing, and debugging	Process evaluation: reflection report; Summative evaluation: project evaluation.
Task 3: The closed reverse task	Task: Make the small rabbit eats off a green cauliflower. Specification: ditto. Troubleshooting: The small rabbit does not stoop when it eats the cauliflower.		
Task 2: The semi-open forward task	Task: Make the big rabbit jumps to the front of red cauliflowers. Specification: The big rabbit's jump is fluent and coordinating, and both feet will swing when jumping.	Sequences, loops, parallelism, modularizing, testing, and debugging	ditto
Task 4: The semi-open reverse task	Task: Make the big rabbit jumps to the front of red cauliflowers. Specification: ditto. Troubleshooting: The big rabbit move to the front of red cauliflowers directly without jumping and swing.		ditto
Task 5: The open task with a creative design report	Task: Design a scenario to describe what is probably happened after the small rabbit became smaller, and need to fill out creative design report before working.	Planning and designing, creative and expressing, abstracting and modeling, testing and debugging, iterative and optimizing, modularizing and reusing, and so on.	Process evaluation: reflection report, creative design report; Summative evaluation: project evaluation.
Task 6: The open task without a creative design report	Task: Design a scenario to describe what is probably happened after the small rabbit became smaller.		Process evaluation: reflection report; Summative evaluation: project evaluation.

Onderbouw voortgezet onderwijs

In totaal bestudeerden we negen studies in de onderbouw van het voortgezet onderwijs.
De studie van Yang en Chang (2013) richtte zich op het effect van de interventie op kritisch denken. Deze kritische denkvaardigheden werden gemeten met de 'Critical Thinking Test-Level I' (Yeh, 2003). Aspecten als inductie, deductie, evaluatie van assumpties werden gemeten.

De studies beschreven in Akcaoglu (2014) en in Akcaoglu en Koehler (2014) geven inzicht in effecten van een interventie op de algemene probleemoplossingsvaardigheden van leerlingen. Daarbij maakten ze gebruik van een gestandaardiseerde test aangeboden door Organisation for Economic Co-operation and Development (OECD, 2003).
Meerbaum-Salant et al. (2013) gebruiken een combinatie van de taxonomie van Anderson et al. (2001) en de SOLO-taxonomie (Biggs 7 Collis, 1982) om vragen te categoriseren voor het meten van computer science kennis en vaardigheden. Van de taxonomie van Anderson gebruiken ze de categorieën 'understanding', 'applying' en 'creating'; deze worden gecombineerd met de SOLO-categorieën 'unistructural', 'multistructural' en 'relational'. Hierdoor kunnen er in totaal negen niveaus geformuleerd worden. Voorbeeldvragen zijn:

- Multistructural / Understanding. What is the difference between the instruction say [hello] and the instruction broadcast [hello]?
- Multistructural / Creating. Add to the following scripts an instruction or instructions that will cause the scenario where the sprites change places to repeat itself five times.
- Relational / Applying. Describe the behavior of the animals and the ball during the
- execution of all the scripts (after clicking on the green flag).

Grover, Pea en Cooper (2015) onderzochten het effect van de interventie op computational kennis. De vragenlijst werd niet beschreven maar was gebaseerd op studies van onder meer Meerbaum-Salant et al. (2010).

De studie van Linn (1985) is onduidelijk als het gaat om het meten van de programmeervaardigheden. Er werd aangegeven dat het een vragenlijst betrof van 23 items met een goede betrouwbaarheid.
In de studie van Pellas en Peroutseas (2016) is nagegaan in hoeverre leerlingen gemotiveerd waren om te werken in een game-gebaseerde omgeving. Ze gebruikten daarvoor de 'User Engagement Scale (Wiebe et al., 2014)'. De vragenlijst bestond uit 31 items met een vijfspuntsschaal (van helemaal eens tot helemaal oneens). De subschalen betroffen: 'focused attention', 'felt involvement', 'novelty', 'endurability', 'aesthetics' en 'perceived usability'. Een voorbeelditem is: 'Playing on this 3D multi-user game-like environment was worthwhile'. De betrouwbaarheid van de schalen was acceptabel.

Door middel van gestructureerde groepsinterviews werd in de studie van Ke (2014) de perceptie van de leerlingen gemeten ten aanzien van wiskunde en het maken van een game. Voorbeeldvragen in het interview waren: What was your game about?; What did you do? What would you add or change about this game?; What do you want your players to learn from this game? Do you think this game will teach? Why or why not?; Do you think knowing math is important for game creation? Why or why not?; How do you feel about your game creation experience? What impressed you most? What frustrated you most?; What have you learned by creating this game? Could you elaborate or give any examples?

De studie van Atmatzidou en Demetriadis (2016) keek naar het effect van een interventie op de computational thinking skills van leerlingen. Ze hanteerden een model met verschillende concepten van computational thinking en formuleerden welke leerdoelen leerlingen zouden moeten bereiken. De vaardigheden werden beoordeeld met een vragenlijst met open vragen. De beoordeling gebeurde met een rubric en een vierpuntsschaal (1 = 'unsatisfactory', 2 = 'quite satisfactory', 3 = 'satisfactory', 4 = 'excellent'). Tabel 9 geeft een overzicht van de gemeten concepten en de te behalen doelen.

Tabel 9 Concepten van Computational Thinking en de te behalen doelen (Atmatzidou en Demetriadis (2016)).

CT skills	Description	Student skills (The student should be able to. . .)
Abstraction	Abstraction is the process of creating something simple from something complicated, by leaving out the irrelevant details, finding the relevant patterns, and separating ideas from tangible details [52]. Wing [2] argues that the essence of CT is abstraction.	1. Separate the important from the redundant information. 2. Analyse and specify common behaviours or programming structures between different scripts. 3. Identify abstractions between different programming environments.
Generalisation	Generalisation is transferring a problem-solving process to a wide variety of problems [38].	Expand an existing solution in a given problem to cover more possibilities/cases.
Algorithm	Algorithm is a practice of writing step-by-step specific and explicit instructions for carrying out a process. Kazimoglu et al. [37] argue that selection of appropriate algorithmic techniques is a crucial part of CT.	1. Explicitly state the algorithm steps. 2. Identify different effective algorithms for a given problem. 3. Find the most efficient algorithm.
Modularity	Modularity is the development of autonomous processes that encapsulate a set of often used commands performing a specific function and might be used in the same or different problems [38].	Develop autonomous code sections for use in the same or different problems.
Decomposition	Decomposition is the process of breaking down problems into smaller parts that may be more easily solved. Wing [2] argues that CT is using decomposition when attacking or designing a large complex task.	Break down a problem into smaller/simpler parts that are easier to manage.

De studie van Berlan en Wilensky (2015) ging na wat het effect was van de interventie op vier concepten: 1) Complex system thinking, 2) Computational thinking, 3) Essay Question on the Relationship Between Classroom Space and 'Robot Space' en 4) Programming skills. Het instrument waarmee ze deze concepten hebben gemeten, bestond uit een beschrijving van een situatie waarover vragen werden gesteld. De vragen werden gescoord op een vierpuntsschaal (0 = niet geantwoord, 1 = incorrect antwoord, 2 = deels correct antwoord, 3 = correct antwoord). Bij computational thinking moesten leerlingen bijvoorbeeld een flowcart interpreteren en verder uitwerken.

Bovenbouw voortgezet onderwijs

Twee studies werden uitgevoerd in de bovenbouw. De studie van Palumbo en Reed 1991 keek naar effect op generieke probleemoplossingsvaardigheden. Daarnaast werd er gekeken naar de leerresultaten tijdens de lessen. Deze werd op drie momenten tijdens de interventie gemeten. Het betrof vooral kennis van leerlingen. Het gebruikte instrument is niet beschreven en ook word niets vermeld over de psychometrische gegevens.

De studie van Liu et al. (2013) heeft het effect van de interventie met behulp van een vragenlijst bestaande uit vijftig meerkeuzevragen gemeten. Deze vragen waren ondergebracht in vier schalen: 1) algemene programmeervragen, 2) ROBOTC syntax-vragen, 3) vragen over fysieke functioneren van een robot, 4) vragen met betrekking tot algoritmisch denken. De betrouwbaarheid van de eerste twee schalen was onder de maat. De schalen 3 en 4 gaven goede betrouwbaarheidscoëfficiënten. In tabel 10 staan voorbeeldvragen die zijn gebruikt om de vier constructen te meten.

Tabel 10 Voorbeeldvragen gehanteerd in de studie van Liu et al (2013).

Schaal	Voorbeeldvragen
General programming	• Special types of variables that allow you to pass data into a function are called (senders/parameters/passers/variables). • Just like task main, functions must have a set of parentheses () and a set of (semi-colons/curly braces/AND operators/brackets). • If the condition of an If statement is true, then all of the code inside of its curly braces will run. True/False. • A decimal number can be stored in which type of variable? (int/char/string/float). • With functions, you write the code once, give it a name and then reuse it as many times as you'd like within a program. True/False.
ROBOTC syntax	• The code "wait1Msec(5000)" tells the robot to wait for (5 s/50 s/one second/half a second). • Which character is used to signify the end of a command in ROBOTC? (comma/period/semicolon/colon). • Before using the encoders in a line-tracking program, their sensor values should be set to (−1/4095/0/100). • Which of the following cannot be used when naming a motor or sensor? (special characters/spaces/ROBOTC reserved words/all of the above). • To make a robot stop, you set its motor values (equal to 0/opposite one another/equal to 63/equal to −1).
Physical robot functioning	• Ignoring drift, when using encoders to control how far a robot travels, slowing the robot down will make it move (a greater distance/the same distance/a shorter distance/an unknown distance). • What can cause one motor to run faster than another on a robot? (construction of the robot/variances between the motors/friction/all of the above). • On the VEX Cortex, the shaft encoders record how many counts or degrees per revolution? (360/100/180/1000). • The VEX Line Tracking sensor works by measuring (reflected light/torque/sound/RPMs). • A properly configured robot that has one motor turned on and the other turned off will perform a (opposite turn/point turn/tank turn/swing turn).
Algorithmic thinking	• The hybrid language halfway between English and the programming language is called (halfcode/hybridcode/prenglish/pseudocode). • Simple behaviours are made up of complex behaviours. True/False. • When comparing the role of the robot vs. the role of the programmer, the pro- grammer's role is to (ignore/enforce/create/carry out) the plan. • Given the program above, the robot will do ___.

4 Conclusie en discussie

Het doel van deze reviewstudie was om inzicht te krijgen in de effecten van programmeeronderwijs op computational thinking (CT) van leerlingen. Een belangrijke aanleiding voor de review zijn de claims die vaak worden toegeschreven aan programmeeronderwijs, met name als het gaat over de bijdragen van programmeren aan generieke vaardigheden, zoals probleemoplosvaardigheden. In navolging van Lye en Koh (2014) werd in deze review CT onderscheiden in *concepts*, *practices* en *perspectives*. De review richtte zich op de effecten van programmeeronderwijs op CT, de condities waaronder die effecten optreden en de wijze waarop deze effecten worden gemeten.

Uit onze review blijkt dat van het kleuteronderwijs tot aan de bovenbouw van het voortgezet onderwijs onderzoek wordt gedaan naar het effect van programmeeronderwijs op CT. Verreweg de meeste studies vonden plaats in de onderbouw van het voortgezet onderwijs, gevolgd door de bovenbouw van het primair onderwijs.

De review laat zien dat er evidentie is voor positieve effecten van programmeeronderwijs op generieke vaardigheden. Van de veertien studies die onderzoek deden naar het effect op generieke vaardigheden werd in elf studies een positief effect gevonden, in acht studies betrof het een positief effect op probleemoplosvaardigheden. Ook vonden we een positief effect van programmeeronderwijs op computer science-concepten en computational practices (vijf studies). In acht studies werd onderzoek gedaan naar het effect van programmeeronderwijs op specifieke aspecten van CT met veelal een positief effect op de onderzochte kennis en vaardigheden. Slechts vier studies deden onderzoek naar de relatie tussen programmeeronderwijs en effecten op de kennis en vaardigheden in andere vakken (zoals rekenen en biologie). Er werd een positief effect gevonden. In geen enkele studie is nagegaan of er sprake is van transfer van de opgedane kennis- en vaardigheden naar andere domeinen. Wel werd in een aantal studies de probleemoplosvaardigheden vastgesteld met behulp van algemene testen, die niet domein- of taakgebonden zijn.

In zes studies werd ook gekeken naar effecten van programmeeronderwijs op motivatie en enthousiasme. In het algemeen zijn leerlingen gemotiveerd om het aangeboden programmeerdonderwijs te volgen en reageren ze enthousiast. Voor veel leerlingen was dit ook de eerste keer dat ze kennis maakten met programmeeronderwijs, dus er kan sprake zijn van een *novelty effect*.

Toch moeten deze bevindingen met de nodige voorzichtigheid worden geïnterpreteerd. In de eerste plaats was het onderzoek meestal geen onderdeel van het reguliere onderwijs. Relatief veel onderzoeken werden uitgevoerd in een specifieke context: buiten schooltijd (bijvoorbeeld Akcaoglu, 2014; Akcaoglu & Koehler, 2014), op privéscholen (bijvoorbeeld Klahr & Carver, 1988; Pardamean et al., 2015) of als onderdeel van een keuzevak (bijvoorbeeld Grover et al., 2015). In de tweede plaats was er relatief vaak

sprake van casestudies of pre-experimentele designs, met een klein aantal leerlingen zonder controlegroep. Van de 26 studies hadden slechts zes studies een design waarin de resultaten van de experimentele groep werden vergeleken met die van een controlegroep. Hieruit blijkt dat het onderzoek naar de effecten van programmeeronderwijs op computational thinking nog in de kinderschoenen staat.

Om te achterhalen onder welke condities de gevonden effecten optreden, is nagegaan wat de kenmerken zijn van het programmeeronderwijs dat werd aangeboden in die studies die een positief effect sorteerden. Uit de bevindingen blijken de volgende kenmerken relatief vaak voor te komen:

- directe instructie bij de introductie van nieuwe concepten
- een opbouw van het onderwijs van eenvoudige concepten (bijvoorbeeld sequentie) naar meer complexe concepten (bijvoorbeeld conditionele en parallelle uitvoering)
- oefenen van nieuwe concepten aan de hand van door de docent geformuleerde opdrachten
- leerlingen werken samen aan de programmeertaak (meestal in tweetallen).

Hoewel de duur van de interventies behoorlijk varieerde werd wel duidelijk dat programmeeronderwijs dat gericht is op CT een bepaalde tijdsinvestering vereist. Uit onze bevindingen bleek niet dat de gekozen programmeeromgeving van belang was voor het al dan niet realiseren van de beoogde effecten. In relatief veel studies werd het onderwijs verzorgd door de onderzoeker of door onderzoeksassistenten. Dit duidt erop dat het vakinhoudelijke kennis en vaardigheden op het gebied van programmeeronderwijs belangrijk is bij de implementatie van programmeeronderwijs in de onderwijspraktijk en dat docenten daarin geschoold moeten worden. Daarin werden geen verschillen gevonden tussen primair en voortgezet onderwijs.

In de bestudeerde studies is gekeken naar effecten van onderwijs op de onderliggende vaardigheden die CT-kenmerken, zoals bij logisch ordenen en debuggen. Ook is gekeken naar effecten van onderwijs op de meer generieke 21e-eeuwse vaardigheden als probleemoplossen. Natuurlijk zijn deze vaardigheden aan elkaar gerelateerd en kan CT gezien worden als een onderdeel van de vaardigheid om problemen op te lossen. In de bestudeerde studies is nagegaan of de effecten zijn gemeten op de specifieke CT-aspecten of op een meer generiek niveau van probleemoplossingsvaardigheden.

Als we kijken naar de wijze waarop het concept CT in de studies werd gemeten, moeten we concluderen dat dit heel divers is. Bij de studies die gedaan zijn in het kleuteronderwijs werd alleen in de studie van Bers et al. (2014) gekeken naar CT-aspecten. De uitvoering van de taken tijdens de interventie werden geobserveerd. In de onderbouw van het primair onderwijs werden verschillende studies uitgevoerd, maar geen van deze studies heeft gekeken naar effecten op CT. Alle studies bepaalden het effect van de interventie op generieke vaardigheden als probleemoplossen en reflectie. Van de studies uitgevoerd in de bovenbouw van het primair onderwijs richtten vier studies zich op het meten van meer generieke vaardigheden als probleemoplossen en mediageletterdheid. Van die overgebleven vijf studies kan worden gezegd dat ze alle op schillende wijze hebben gemeten. De studie van Moreno et al. (2015) evalueerde de gegeven workshop met een vragenlijst. De studie van et al. (2014) analyseerden de game patterns van leerlingen. De studie van Klahr en Carver (1988) was gericht op het meten van de debugging-vaardigheden en gebruikten daarvoor taakjes. De studie van Sáez-López et al. (2016) hanteerde een vragenlijst om CT concepts en practices te meten. De studie van Zhong et al. (2016) was gericht op het meten van CT en biedt vanuit een theoretisch kader een goede opzet ten aanzien van het meten van CT. Deze inzichten ten aanzien van soorten taken kunnen worden gebruikt om verder te onderzoeken hoe CT kan worden gemeten en om te komen tot een gedegen meetinstrument.

Van de negen studies die zijn uitgevoerd in de onderbouw van het voortgezet onderwijs zijn er eigenlijk maar twee studies die specifiek CT meten (Atmatzidou en Demetriadis, 2016; Berlan en Wilensky, 2015). De andere studies meten effecten op generieke vaardigheden, percepties van leerlingen en attitude. Atmatzidou en Demetriadis (2016) gebruikten een rubric om de concepten die leerlingen hanteerden te meten. Deze was specifiek voor de ontworpen interventie. Berlan en Wilensky (2015) stelden vragen bij beschreven situaties. Over de betrouwbaarheid en validiteit van de meetinstrumenten is niet veel te zeggen. In de bovenbouw van het voortgezet onderwijs zijn twee studies uitgevoerd. Alleen in de studie van Liu et al. (2013) werd CT gemeten. Men deed dat aan de hand van een vragenlijst met verschillende schalen. Deze waren echter niet alle even betrouwbaar.

Over het algemeen kan worden gesteld dat er in de onderzoeken op verschillende manieren is gemeten. Er moet nog verder onderzoek worden gedaan om voor verschillende doelgroepen te komen tot het meten van CT. De studie van Zhong et al. (2016) geeft goede aanknopingspunten om vanuit een theoretisch perspectief CT uit te werken richting betrouwbaar en valide meetinstrumenten voor leerlingen in verschillende leeftijdscategorieën.

5 Referenties

*Akcaoglu, M. (2014). Learning problem-solving through making games at the game design and learning summer program. *Educational Technology Research and Development, 62*(5), 583-600.

*Akcaoglu, M., & Koehler, M. J. (2014). Cognitive outcomes from the Game-Design and Learning (GDL) after-school program. *Computers & Education, 75*, 72–81.

*Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661-670. doi: 10.1016/j.robot.2015.10.008.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and What is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.

*Berland, M., & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology, 24*, 1– 20^.

*Bers, M. I., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72,* 145–157.

Brand-Gruwel, S., & Stadtler, M. (2011). Solving information-based problems: Evaluating sources and information. *Learning and Instruction, 21,* 175-179. doi:10.1016/j.learninstruc.2010.02.008.

*Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognitions. *Journal of Educational Psychology*, *76*, 1051–1058.

*Ennis, D. (1994). Combining problem solving and programming instruction to increase the problem solving abilities in high school students. *Journal of Research on Computing in Education*, *26*(4), 488–496.

*Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 year old kindergarten children in a computer programming environment: a case study.

Computers & Education, 63, 87-97. Retrieved from http://dx.doi.org/10.1016/j.compedu.2012.11.016.

Fisser. P., & Strijker, A. (2016, 5 februari). *Computational thinking in het kader van 21e eeuwse vaardigheden.* Paper gepresenteerd op de VELON-conferentie, Brussel.

Grover, S., & Pea, R. (2013). Computational thinking in K-12, a review of the state of the field. *Educational Researcher, 42*(1), 38-43.

*Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199-237.

*Jenson, J., & Droumeva, M. (2016). Exploring media literacy and computational thinking: a Game Maker curriculum study". *Electronic Journal of e-Learning, 14*(2), 111-121.

Jeuring, J., Corbalan, G., Van Es, N., Van Leeuwestein, H., & Van Montfort. J. (2016). *Leren programmeren in het PO – een literatuurreview*. Literatuurreview uitgevoerd in opdracht van de Kennisrotonde, het online loket voor de beantwoording van actuele kennisvragen uit het onderwijs.

*Kalelioglu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*(3), 200–210.

*Kalelioglu, F., & Gülbahar, Y. (2014). The effect of teaching programming via scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education, 13*(1), 33–50.

*Kazakoff, E., Sullivan, A., & Bers, M. (2012). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245–255.

*Ke, F. (2014). An implementation of design-based learning through creating educational computer games: a case study on mathematics learning during design and computing. *Computers & Education, 73*(1), 26–39. http://dx.doi.org/10.1016/j.compedu.2013.12.010.

*Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology, 20*(3), 362–404.

*Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher, 14*(5), 14–29.

*Liu, A. S., Schunn, C. D., Flot, J., & Shoop, R. (2013). The role of physicality in rich programming environments. *Computer Science Education, 23*(4), 315-331.

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

*Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education, 23*(3), 239-264.

*Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED, Revista de Educación a distancia, 46*.

*Palumbo, D.B. & Reed, W.M. (1991). The effect of BASIC programming language instruction on high school students' problem solving ability and computer anxiety. *Journal of Research on Computing in Education, 23*, 343-345.

*Pardamean B., Suparyanto T., & Evelyn (2015). Improving problem-solving skills through Logo programming language. *The New Educational Review*, *41*, 52–64.

*Pellas, N., & Peroutseas, E. (2016). Gaming in second life via Scratch4SL: Engaging high school students in programming courses. *Journal of Educational Computing Research*, *54*(1), 108-143. doi:10.1177/0735633115612785

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

Sengupta, P., Kinnebrew, J., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351–380.

The College Board (2013). *AP computer science principes draft curriculum framework*. New York, NY: College Board. Retrieved from http://www.csprinciples.org/home/about-the-project.

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

*Werner, L., Denner, J., & Campe, S. (2014). Children programming games: A strategy for measuring computational learning. *ACM Transactions on Computing Education*, *14*, 1-22.

Wing, J. (2006). Computational thinking. *Communication in ACM, 49*(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 366*(1881), 3717–3725.

Wolz, U., Stone, M., Pearson, K., Pulimood, S., & Switzer, M. (2011). Computational thinking and expository writing in the middle school, *ACM Transactions on Computing Education, 11*, 2, article 9.

*Yang, Y. C., & Chang, C. L. (2013). Empowering students through digital game authorship: Enhancing concentration, critical thinking, and academic achievement. *Computers & Education*, *68*, 334–344.

*Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research*, *53*(4), 562-590. doi:10.1177/0735633115608444

*) Betreft interventies

Bijlage 1 Exacte zoektermen

Set 1: Computational thinking en gerelateerde concepten

computational thinking OR algorithmic thinking OR algorithmic skill* OR analytical thinking OR problem solving skill* OR mathematical thinking OR computationalist thinking OR logical thinking OR engineering thinking OR design thinking OR critical thinking OR analytical thinking OR problem solving skill* OR mathematical thinking OR computationalist thinking OR logical thinking OR engineering thinking OR design thinking OR critical thinking OR analytical thinking OR problem solving skill* OR mathematical thinking OR computationalist thinking OR logical thinking OR engineering thinking OR design thinking OR critical thinking

Set 2: Programmeren en gerelateerde concepten

informatics education OR computer science education OR computer science OR computer programming OR programming skill* OR computing OR computer education code OR coding OR programming OR coding skills OR informatics OR code OR coding OR programming OR coding skills OR informatics OR code OR coding OR programming OR coding skills OR informatics

Set 3: Doelgroep

elementary education OR secondary education OR elementary school* OR primary education OR intermediate grades OR middle school* OR junior high school* OR secondary school* OR high school* OR primary school* OR K-12 OR K12 OR junior high* OR highschool* OR preuniversity OR pre-university OR grade 1 OR grade 2 OR grade 3 OR grade 4 OR grade 5 OR grade 6 OR grade 7 OR grade 8 OR grade 9 OR grade 10 OR grade 11 OR grade 12 OR elementary school* OR primary education OR intermediate grades OR middle school* OR junior high school* OR secondary school* OR high school* OR primary school* OR K-12 OR K12 OR 1st-grade* OR first-grade* OR grade 1 OR grade one OR 2nd-grade* OR second-grade* OR grade 2 OR grade two OR 3rd-grade* OR third-grade* OR grade 3 OR grade three OR 4th-grade* OR fourth-grade* OR grade 4 OR grade four OR 5th-grade* OR fifth-grade* OR grade 5 OR grade five OR 6th-grade* OR sixth-grade* OR grade 6 OR grade six OR intermediate general OR 7th-grade* OR seventh-grade* OR grade 7 OR grade seven OR 8th-grade* OR eight-grade* OR grade 8 OR grade eight OR 9th-grade* OR ninth-grade* OR grade 9 OR grade nine OR 10th-grade* OR tenth-grade* OR grade 10 OR grade ten OR 11th-grade* OR eleventh-grade* OR grade 11 OR grade eleven OR 12th-grade* OR twelfth-grade* OR grade 12 OR grade twelve

Bijlage 2 Programmeeronderwijs: effectieve interventies

(Tabellen vanaf volgende pagina)

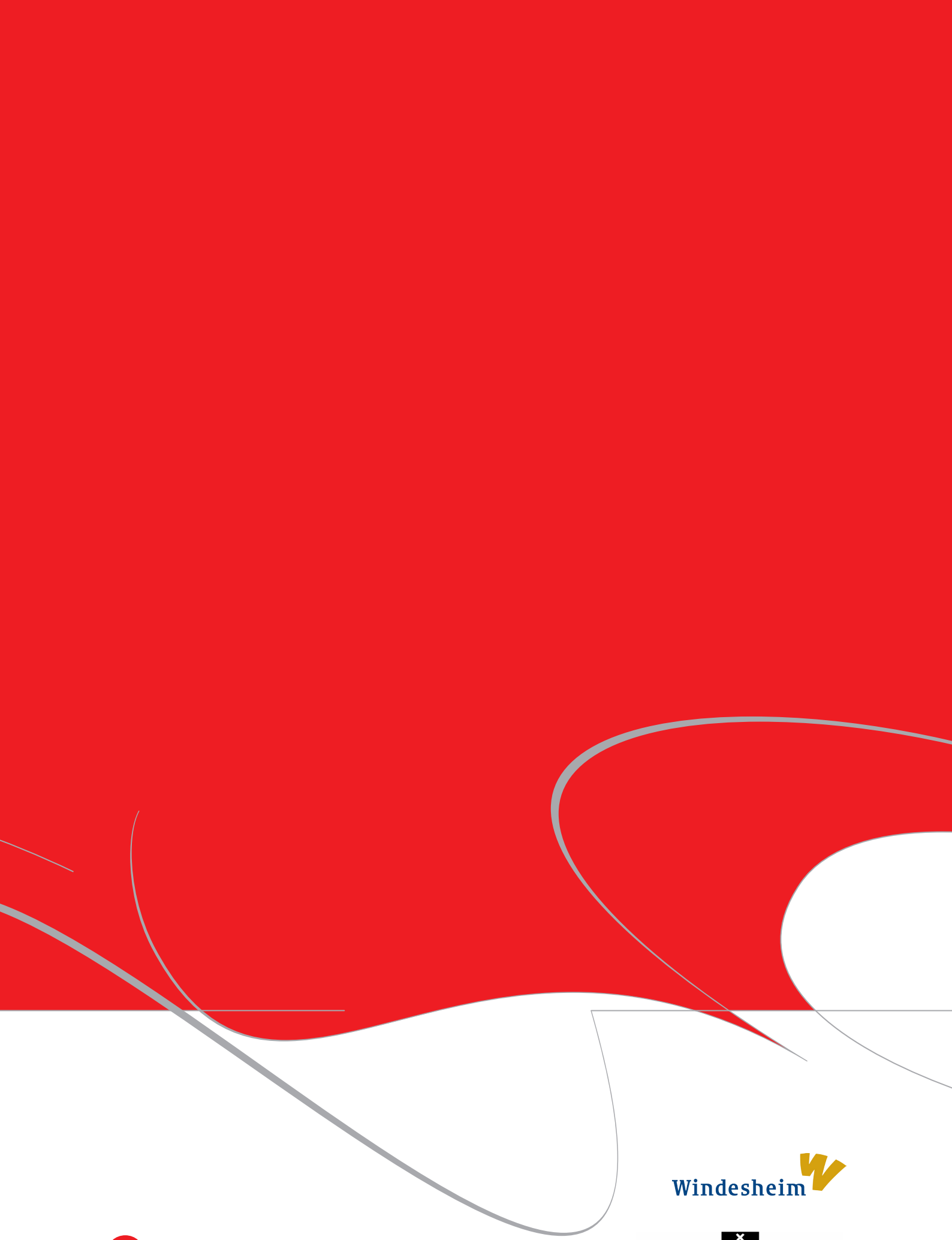
Auteur	*Uitkomstvariabele*	*programmeeromgeving*	*duur*	*inhoud*	*Leerlingactiviteiten*	*Leeromgeving (o.a. docentrol /groeperingsvorm)*
Kleuters						
Fessakis et al.	Generieke vaardigheden: Probleemoplossvaardigheden Vakinhoudelijke vaardigheden: Rekenvaardigheden	Logo met beperkt aantal commando's	7 activiteiten (duur onbekend)	Gezamenlijke start activititeit (zonder computer) 7 computeractiviteiten gericht op het vinden van de weg en het vinden/oplossen van fouten in een doolhof	Eén kind voert de opdracht uit en de andere kinderen praten mee o.l.v. de leerkracht	De docent stimuleert en ondersteunt de leerlingen bij de aanpak en het oplossen van het problem; ze coördineert de klassikale activiteit (beurten geven). Klassikaal met behulp van IWB
Bers et al.	Specifieke CS-concepten/ practices	Robot constructie kits + CHERP (Creative Hybrid Environment for Robotics Programming)	6 sessies van 60-90 minuten	6 lessen+ eindproject: kennis maken met het ontwerpproces (plannen, testen, verbeteren) - offline (1); robotica: leren hun eigen bewegende robot ontwerpen en bouwen (2); programmeren van robot (3); herhaal instructies (loops) (4); sensoren (5); voorwaardelijke instructies (6) en eindproject. Het curriculum omvat liedjes, spelletjes en vrij spelen met de robot om een speelse leeromgeving voor kinderen te creëren.	De kinderen werken zelfstandig aan de opdracht. Aan het begin van de les zingen leerlingen een lied of spelen een spel passend bij de activiteit. Aan het eind van de les delen zij hun voortgang, hun vragen en hun aanpak o.l.v. de docent.	Docent introduceert kennis en activiteiten per sessie en leidt de groepsdiscussie aan het eind. Docent heeft hulp van assistenten om de leerlingen te begeleiden Start en slot van de sessie is klassikaal. Daar tussen in werken kinderen individueel aan hun robot.
Kazakoff et al.	Specifieke CS-concepten/ practices	Zie Bers et al.	Te weinig informatie	Te weinig informatie	Te weinig informatie	Te weinig informatie
Primair onderwijs onderbouw						
Clements & Gullo	Generieke vaardigheden: creatief denken; reflectieve vaardigheid	Logo	12 weken (2 per week)	1. introductie programmeeromgeving (2 sessies); 2. procedures om geometrische vormen te ontwerpen (4); 3. schrijven en combineren van procedures (8); 4. plannen/maken superprocedures (6)	1. kinderen plannen op papier; 2. opsplitsen geometrische vorm in delen (huis = vierkant + driehoek) en labelen; 3. kinderen verbeteren fouten in procedure geschreven door onderzoeker 4. kinderen schrijven zelf een programma; dit wordt steeds complexer; 5. kinderen leren hoe deelprocedures in superprocedure	Docentrol wordt ingevuld door onderzoeker; 2-3 kinderen met de onderzoeker

Primair onderwijs bovenbouw						
Klahr & Carver	Generieke vaardigheden: probleemoplos-vaardigheden CS-concepten: debug-gen	Logo	25 weken, 2 keer 40 min. per week	Een les gewijd aan de introductie van nieuwe concepten (uiteenrafelen van het probleem, hergebruik van procedures, samenstellen van deelprocedures). Na 6-8 uur programmeerervaring één expliciete les waarin een strategie wordt aangeleerd om fouten op te sporen en op te lossen; Leerlingen krijgen drie toetsen waarin zij in een gegeven programma's fouten moeten opsporen en oplossen	Leerlingen werken naast de opdrachten gerelateerd aan de toetsen aan eigen projecten	De docent introduceert nieuwe concepten; De rol van de docent is daarna beperkt tot een minimum; Groepjes van 2 Concepten en commando's op posters aan de muur
Pardamean et al.	Generieke vaardigheden: Probleemoplos-vaardigheden	Logo	16 sessies van 40 minuten (8 weeks)	Introductie in de taal, het ontwerpen van simpele en meer complexe procedures voor het tekenen van geometrische vormen.	Leerlingen werken aan taken. Aan het eind van de les krijgen zij een toets.	De docent is coach en informatiebron Leerlingen werken vaak samen aan taken; de toetsen zijn individueel
Saez	CS-concepten/ practices Vakinhoudelijke kennis: kunstgeschiedenis	Scratch	20 sessies van één uur gedurende twee jaar	Te weinig informatie	Te weinig informatie	Te weinig informatie
Jenson & Droumeva	CS-concepten	Game maker studio	6 sessies van elk 1,5 uur	Geleidelijke introduceren van de concepten (variabelen, operaties, functies, condities). Elk concept wordt gevolgd door oefenen en toepassen van het concept. Geleerde concepten worden herhaald en versterkt.	De leerlingen werken aan hun eigen spel. De leerlingen moeten eerst zelf een oplossing proberen te vinden, met behulp van de hulp die het programma biedt, voordat ze worden geholpen.	De onderzoekers verzorgen de lessen Leerlingen werken in tweetallen
Moreno-Leon et al.	Computational practices	Scratch en Dr. Scratch	Te weinig informatie	Te weinig informatie	Te weinig informatie	Te weinig informatie

Voortgezet onderwijs onderbouw						
Yang & Chang	Generieke vaardigheden: kritisch denken Vakinhoudelijke kennis: biologie	Digital Game Authorship (RPG maker)	19 weken (totale interventie, waarvan 14 weken onderwijs- de rest testen(1 * per week)	1. instructie in kritisch denken (3 weken); 2. onderwijs in biologie (11); parallel daaraan: 3. introductie in gebruik software (6) en ontwikkelen van de game (5) in zes stappen: introductie, taakbeschrijving, hints, een quiz, een conflict en de conclusie.	De leerlingen maken digitale spellen; Er is een gedeelde verantwoordelijkheid voor het eindproduct. Na het ontwerp werken de leerlingen aan karakters, achtergronden, routes, conflicten, quizvragen, enzovoort. In de laatste week worden de spellen geëvalueerd door mede leerlingen en bediscussieerd.	Rol docent niet expleciet beschreven; Groepen van 5-6 leerlingen met wisselende leiderschapsrol;
Ke	Generieke vaardigheden: Probleemoplosvaardigheden Vakinhoudelijke vaardigheden: wiskunde	Scratch	6 weken 1 uur per week	Voor de interventie: leerlingen spelen wiskunde games + leerlingen kregen een workshop (3uur) om scratch te leren kennen (incl. scratch archief); identificatie van wiskunde concepten; ontwerp game op papier; computer-assisted game design & testing	Opdracht ontwerp een mini wiskunde/reken game voor jongere leeftijdsgenoten	Promovendi hebben een coachende rol: beantwoorden vragen, geven feedback en zijn mentor Ontwerpgroepen van 6-7 personen
Atmatzidou & Demetriadis	Generieke vaardigheden: Probleemoplosvaardigheden CS-concepten: algoritmes, moduleren, decompositie	Lego mindstorms	11 sessies 2 uur per week	Introductie in robotics; algoritme(sessie 1); sequentie en loop; abstractie en generalisatie (2):control structuur- modulariteit en decompositie (3) toepassen alle concepten (4 en verder): herbruikbare procedures (5-6); variabelen rekenkundige operatoren (7-8); daarna steeds complexere problemen	Programmeeropdrachten die steeds moeilijker worden; met meer gebruik van de mogelijkheden van het materiaal;	Promovendi hebben de rol van trainer. Groepen leerlingen (omvang niet bekend)

Berland & Wilensky	Complexe systemen	VBOT (grafisch georiënteerde programmeertaal	5 achtereenvolgende dagen	Instructie in de VBOT syntax (dag 1-30 minuten; dag 2 en 3- 20 minuten); studenten voeren elke dag een bepaalde opdracht uit: 1. circuit met licht sensoren en motor; 2. vbot sensoren, motor en wiskunde blok; 3; logische operatoren; 4 en 5: (virtueel of fysiek) soccer spel	Leerlingen voeren opdrachten uit	Co-teaching van onderzoeker en docent. Interventies zijn minimaal na initiële instructie. Leerlingen werken individueel
Akcaoglu	Generieke vaardigheden: Probleemoplosvaardigheden	Kodu	9 sessies (extra curriculair)	Game design: introductie in software, game design en programmeren (3 sessies); probleemoplossen: (a) introductie in probleem; oplossen van probleem; ontwerp van simulatie voor het probleem in de game design omgeving; (3) toubleshooting: herstellen van niet werkende games (2); eigen ontwerp (1)	Game design: Leerlingen programmeren programma's en maken stroomschema's om de complexiteit van hun programma te begrijpen. Probleemoplossen: leerlingen analyseren het probleem, plannen een oplossing ontwerpen een gesimuleerde oplossing, die evalueren ze en reflecteren daarop. Leerlingen identificeren fouten in niet werkende games en herstellen die	Instructeur is begeleider en rolmodel
Akcaoglu & Koehler	Generieke vaardigheden: Probleemoplosvaardigheden	Kodu	5 sessies, totaal 15 uur (extra curriculair	Zie: Akcaoglu	Zie: Akcaoglu	Turkije: 1 instructeur US: 1 instructeur; 2 assistenten De instructor is begeleider en rolmodel; assistenten: organisatie en klassenmanagement
Linn	Computational practices	Basic	12 weken	Te weinig informatie	Te weinig informatie	Te weinig informatie

Meerbaum et al..	Specifieke CS-concepten/ practices	Scratch	Niet bekend	Methode: georganiseerd rond de te onderscheiden concepten en een specifiek project per concept; specifieke programmeer- instructies worden just-in-time geïntroduceerd; het opleuken (geluid; aankleding) van het ontwerp wordt ontmoedigd.	Leerlingen werken aan specifieke deeltaken in het kader van het project.	Docenten gebruikten de methode ontworpen door de onderzoekers
Grover et al.	Specifieke CS-concepten/ practices	Scratch	7 weken, 4 dagen per week, sessies van 55 minuten	1. Computing is overal; 2. Wat zijn algoritmes en programma's; 3. Control flow: iteratie en loops; 4. Representatie van informatie: data en variabelen; 5. booleaanse logica en geavanceerde loops; 6. conditionele instructies; eindproject	Leerlingen werden aangemoedigd eerst te (hardop) denken dan te doen aan de hand van voorbeelden en gebruik te maken van pseudocode om het proces om de oplossing uit te werken Eindproject naar eigen keuze	Onderzoeker heeft rol docent Eindproject alleen of in tweetallen In studie 2 is het programma geïntegreerd in een MOOC: met 60 video's (1–5 min in lengte), 10 testjes en activiteiten die individueel of samen konden worden uitgevoerd.
Voortgezet onderwijs bovenbouw						
Palumbo & Reed	Generieke vaardigheden: probleemoplosvaardigheden	Basic	15 weken	1. declaratieve kennis over basic programmeren; 2. declaratieve kennis en procedurele kennis leren integreren; 3. declaratieve en procedurele kennis verder uitbouwen;	1. commando's leren; 2. voorgeschreven problemen programmeren; 3. eigen geformuleerde problemen programmeren	De docent heeft de instructierol
Liu et al.	Computational practices	Robot Virtual Worlds (RVW); ROBOTCprogramming language	9 units	Onderwerpen die aan de orde komen: leren kennen van de software, leren programmeren van de fysieke robot - basics (bijv. Vooruitgaan, verschillende snelheden) en meer geavanceerde concepten (bijvoorbeeld functies en sensoren).		Elke unit heeft: video lectures, op papier gebaseerde activiteiten, online oefenen, quizzen en uitdagingen. Die zowel gesimuleerd kunnen worden of uitgevoerd door de fysieke robot.



Windesheim

Welten-instituut
Onderzoekscentrum voor leren, doceren en technologie

Universiteit van Amsterdam