MASTER'S THESIS

Het effect van visuele programmeeromgevingen op de ontwikkeling van computational thinking skills en het zelfregulerend vermogen bij bovenbouwleerlingen in het basisonderwijs.

Vansteenkiste, Maxim

Award date:
2023

Open Universiteit

Het Effect van Visuele Programmeeromgevingen op de Ontwikkeling van Computational Thinking Skills en het Zelfregulerend Vermogen bij Bovenbouwleerlingen in het Basisonderwijs

The Effect of Visual Programming Environments on the Development of Computational Thinking and Self-Regulated Learning Skills in Upper Primary School Children

Maxim Vansteenkiste

Master Onderwijswetenschappen, Open Universiteit

E-mailadres: vansteenkiste.maxim@outlook.com

Cursuscode en cursusnaam: OM 9906 - Masterthesis

Naam begeleider: Dr. Nardie Fanchamps

Woordenaantal: 7659

Datum: 22/02/2023

Samenvatting

Om computational thinking (CT) te ontwikkelen, kan uit verschillende visuele programmeeromgevingen worden gekozen. Naast een focus op het ontwikkelen van specifieke vaardigheden gelinkt aan CT, kunnen andere pedagogische vaardigheden een even belangrijke rol spelen. Het doel van dit onderzoek is om bij vierenzeventig leerlingen van de bovenbouw van het basisonderwijs in België na te gaan wat het effect is van leren programmeren in een visuele programmeeromgeving op de ontwikkeling van CT, en in welke mate dit effect wordt beïnvloed door het zelfregulerend vermogen. Om dit te onderzoeken, werd een kwantitatief quasi-experimenteel onderzoeksontwerp gebruikt met behulp van twee interventiegroepen die leren programmeren in een visuele programmeeromgeving met een tastbare of beeldschermoutput. Om het verschil in de ontwikkeling van CT en de invloed van zelfregulerend vermogen te bepalen, werd een pretest-posttest design gebruikt. De resultaten laten zien dat er significante verschillen in CT-ontwikkeling merkbaar zijn tussen de twee programmeeromgevingen, maar dat het zelfregulerend vermogen van leerlingen een zwakke maar geen significante invloed had op de ontwikkeling van CT. Op basis van de verkregen resultaten wordt aanbevolen om nader te onderzoeken welk belang kan worden toegekend aan het reguleren van gedachten, gevoelens, vermogens en gedrag, en dat samenwerken met de omgeving en omgaan met feedback bij het ontwikkelen van CT.

Keywords: programmeren, computational thinking, zelfregulerend vermogen, tastbare output, visuele output

Abstract

To develop computational thinking (CT), a variety of programming environments can be selected from. Apart from a focus on developing specific skills, other pedagogical conditions can play an equally important role. The aim of our study is to investigate, among seventy-four upper grades primary school students in Belgium, the effect of learning to program in a visual programming environment on the development of CT, and to what extent this effect is influenced by self-regulatory ability. To investigate this, a quantitative quasi-experimental research design was adopted using two intervention groups that applied either a visual programming environment with tangible output or a visual programming environment with on-screen output. To determine the difference in the development of CT and the influence of self-regulatory ability, a pretest-posttest design was used. The initial indications show that significant differences in CT development are noticeable between the two programming environments, but that learners' self-regulatory ability had a weak but no significant impact on the development of CT. Based on the results obtained, it is recommended to further investigate the importance that can be attributed to regulating thoughts, feelings, abilities and behaviour in the development of CT.

Keywords: programming, computational thinking, self-regulated learning, tangible output, on-screen output

Inhoud

De Invloed van Visuele Programmeeromgevingen op de Ontwikkeling van Computationel Thinking Skills en de Invloed van het Zelfregulerend Vermogen bij Bovenbouwleerlingen in het Basisonderwijs

1. Inleiding

Door de toegenomen technologisering, informatisering en globalisering is onze samenleving in de 21ste eeuw sterk veranderd (Saavedra & Opfer, 2012; Siddiq et al., 2017). Als gevolg van deze toenemende maatschappelijke complexiteit is het belang van conceptuele en metacognitieve vaardigheden op het domein van communicatie, samenwerking, sociaal-cultureel bewustzijn en digitale geletterdheid sterk gegroeid (Thijs et al., 2014). Het gevolg hiervan is dat ook in het onderwijs voldoende aandacht moet besteed worden aan de ontwikkeling van deze vaardigheden, teneinde leerlingen kunnen uitgroeien tot volwaardige participanten van de 21ste eeuwse samenleving (Bellanca, 2010; Griffin & Care, 2014; Voogt & Roblin, 2010). Uit onderzoek (Schleicher, 2012; Voogt & Roblin, 2010) blijkt dat heel wat leraren het niet eenvoudig vinden om deze vaardigheden vorm te geven in de klaspraktijk. Onder meer vaardigheden met betrekking tot digitale geletterdheid en de ontwikkeling van het zelfregulerend vermogen krijgen nog te weinig aandacht in het huidige onderwijs (Thijs et al., 2014). Dit kan tot gevolg hebben dat leerlingen deze vaardigheden onvoldoende ontwikkelen met als gevolg ze moeilijkheden zullen ondervinden om deel te nemen aan een maatschappij die de laatste jaren sterk geënt is op digitalisering (Bellanca, 2010).

Eén van de fundamentele vaardigheden van digitale geletterdheid is computational thinking (CT). Computational thinking is het denkproces waarbij problemen worden geherformuleerd op een manier waardoor deze door Informatie en Communicatie Technologie (ICT)-toepassingen kunnen worden opgelost (Wing, 2011). Het stimuleert hierbij het vermogen om te redeneren, abstraheren en problemen op te lossen (Bastiaensen & De

Craemer, 2017; Grover & Pea, 2013; Wing, 2006). Een geschikte leeractiviteit om computational thinking te ontwikkelen is programmeren (Lye & Koh, 2014; Voogt, Brand-Gruwel & Van Strien, 2017). Om de basisbeginselen van het programmeren aan te leren wordt in de basisschool vaak gebruikgemaakt van visuele programmeeromgevingen omwille van onder meer hun toegankelijkheid voor beginners (Lye & Koh, 2014; Rodríguez-Martínez et al., 2020). Programmeren in visuele programmeeromgevingen wordt gekenmerkt door het slepen en plaatsen van visuele elementen, zoals blokken en pijlen. Dit in tegenstelling tot bijvoorbeeld tekstueel programmeren waarbij de instructies moeten worden ingevoerd aan de hand van tekst. Kennis van een programmeertaal is bij visueel programmeren niet vereist wat de kans op invoerfouten significant kleiner maakt (Bocconi et al., 2016; Kong & Wang, 2019). De output en het gevolg van de acties van de leerling zijn eveneens meteen zichtbaar op een beeldscherm of via een tastbaar object. De verschillen tussen visuele programmeeromgevingen met een beeldschermoutput en een tastbare output met betrekking tot het effect op de ontwikkeling van CT-skills is nog niet eenduidig.

Uit onderzoek (Zheng et al., 2018) blijkt dat naast de soort viusele programmeeromgeving ook het zelfregulerend vermogen van leerlingen een invloed kan hebben op de mate waarin ICT-vaardigheden verworven worden in digitale omgevingen. Lee et al. (2008) stellen dat leerlingen die moeilijkheden hebben om zichzelf te reguleren meer moeilijkheden zullen ondervinden in digitale leeromgevingen. Zelfregulatie houdt in dat een leerling in staat is om zelfstandig te handelen en verantwoordelijkheid op te nemen bij het uitvoeren van leertaken (Bjork et al., 2013; Pintrich, 2000; Vandevelde et al., 2013; Zimmerman, 2000). Om zelfregulerende vaardigheden te ontwikkelen bij leerlingen is er nood aan ondersteuning en een betekenisvolle context waarin zelfregulatie geoefend kan worden (Paris & Newman, 1990;Van Merriënboer, 2017; Zimmerman & Schunk, 2001). Een visuele programmeeromgeving kan hierbij een mogelijke faciliterende rol spelen door zijn focus op

realistische leeractiviteiten, opbouw in complexiteit en de mogelijkheid om te differentiëren in de aangeboden begeleiding (Bocconi et al., 2016). Hoewel er in de literatuur een verband is tussen programmeren en de ontwikkeling van het zelfregulerend vermogen (Ramalingam et al., 2004) is het nog niet eenduidig of programmeren in een visuele programmeeromgeving een invloed heeft op de ontwikkeling van het zelfregulerend vermogen van leerlingen (Fanchamps et al., 2019; McWhorter, 2008; Prather et al., 2020). Daarnaast blijft ook de vraag in welke mate het zelfregulerend vermogen van de leerling een invloed heeft op het verwerven van computational thinking skills in een visuele programmeeromgeving.

Het doel van dit kwantitatieve onderzoek is dan ook om na te gaan welk effect leren programmeren in een visuele programmeeromgeving met een beeldscherm- of viusele output heeft op de ontwikkeling van CT- skills en het zelfregulerend vermogen. Daarnaast willen we nagaan of er een verband is tussen de ontwikkeling van CT-skills en de ontwikkeling van het zelfregulerend vermogen bij bovenbouwleerlingen van de basisschool (9-12j.).

1.1 Theoretisch Kader

1.1.1 Vaardigheden van de 21ste eeuw

Globalisatie, snelle veranderingen op het gebied van informatie en communicatie technologie (ICT) en de behoefte aan innovatie hebben ervoor gezorgd dat onze maatschappij in de 21ste eeuw steeds complexer wordt (Saavedra & Opfer, 2012; Siddiq et al., 2017). Het onderwijs, werkveld en maatschappelijke instanties verwachten steeds meer dat werknemers beschikken over vaardigheden die het mogelijk maken om te kunnen gaan met complexe en onvoorspelbare situaties (Levy & Murnane, 2012). De vaardigheden die belangrijk zijn om te functioneren in en bij te dragen aan de huidige maatschappij worden de 21ste eeuwse vaardigheden genoemd (Care et al., 2012; Siddiq et al., 2017). Deze vaardigheden en de daaraan gekoppelde kennis, inzichten en attitudes zijn generiek en vakoverstijgend (SLO, 2019a).

Welke vaardigheden dit precies zijn is onderwerp van discussie en er bestaan dan ook meerdere theoretische modellen die deze vaardigheden definiëren (Saavedra & Opfer, 2012; Siddiq et al., 2017; SLO, 2019a). De organisatie *Partnership for 21st Century Skills* definieert elf 21ste eeuwse vaardigheden en deelt deze onder in drie types: leervaardigheden (creativiteit en innovatie, kritisch denken en probleemoplossend vermogen, communicatie en samenwerking), geletterdheid (informatie geletterdheid, media geletterdheid en ICT-geletterdheid) en levensvaardigheden (flexibiliteit en aanpassingsvermogen, zelfsturing, sociale en cross-culturele vaardigheden, productiviteit en aansprakelijkheid, leiderschap en verantwoordelijkheid) (van Laar et al., 2020). Het internationale onderzoeksproject *Assessment and Teaching of 21st Century Skills* (ATC21S) definieert tien vaardigheden die worden opgedeeld in vier categorieën: manieren om te denken (creativiteit en innovatie, kritisch denken, probleemoplossend vermogen en besluitvaardigheid, leren leren en metacognitie), manieren om te werken (communicatie, samenwerking), omgang met hulpmiddelen (informatie geletterdheid, ICT-geletterdheid) en leven in de wereld (burgerschap, levens en beroepsvaardigheden, persoonlijke vaardigheden en sociale verantwoordelijkheid) (Binkley et al., 2012). De *Organisation for Economic Co-operation and Development* (OECD) stelt dat alle 21ste eeuwse vaardigheden kunnen ingedeeld worden in de volgende categorieën: informatie, communicatie, ethiek en sociale impact (Siddiq et al., 2017). Hoewel verscheidene organisaties verschillende 21ste eeuwse vaardigheden hebben gedefinieerd, zijn er toch heel wat overeenkomsten tussen de theoretische modellen. Voogt en Roblin (2012) hebben acht modellen met elkaar vergeleken en vonden dat volgende vaardigheden in nagenoeg alle modellen voorkomen: digitale geletterdheid, samenwerking, communicatieve vaardigheden, productiviteit, creativiteit, sociaal-cultureel bewustzijn, kritisch denken en probleemoplossende vaardigheden.

Om deze 21ste eeuwse vaardigheden naar de onderwijscontext te vertalen werden deze herleid tot een model met zeven vakoverstijgende competenties: samenwerken, digitale geletterdheid, probleemoplossend denken en handelen, creatief denken en handelen, zelfregulering, socio-culturele vaardigheden en communiceren (Kennisnet, 2016; SLO, 2019a). In 2014 bleek echter dat in het onderwijs deze competenties slechts in beperkte mate worden geïntegreerd in de leerplannen en methodes. Specifiek waren de vaardigheden probleemoplossend denken en digitale geletterdheid nog onvoldoende uitgewerkt. Om deze reden werd in 2015 het model verder geconcretiseerd waarbij digitale geletterdheid werd opgesplitst in mediawijsheid, informatievaardigheden en computational thinking. Figuur 1 geeft het huidige model weer met de 21ste eeuwse vaardigheden die door Kennisnet (2016) en SLO (2021) worden gedefinieerd voor het onderwijs.

Figuur 1. Model 21ste-eeuwse vaardigheden (SLO, 2021)

1.1.2 Computational thinking

Door de snelle veranderingen in de 21ste eeuw op het gebied van ICT is het belang om computertechnologie efficiënt en effectief te leren gebruiken sterk toegenomen (SLO, 2020a). Leren begrijpen hoe informatie tot stand komt en filteren, inzicht krijgen in algoritmes en procedures, patronen herkennen, oplossingen kunnen bedenken en uitvoeren zijn enkele voorbeelden van vaardigheden die hierbij onontbeerlijk zijn (Bastiaensen & De Craemer, 2017; SLO, 2020a).

Computationeel denken werd reeds lang voor het uitvinden van de computer toegepast. Zo beschreven de Babyloniërs in het jaar 1800 voor onze tijdsrekening procedures voor het oplossen van wiskundige problemen (Denning & Tedre, 2019). De term computational thinking werd voor het eerst gebruikt in 1980 door Seymour Papert toen hij de educatieve programmeertaal LOGO ontwikkelde (Kennisnet, 2016). Papert (1980) kon aan de hand van zijn programmeertaal aantonen dat computertechnologie gebruikt kan worden als een hulpmiddel om het leren en denken van kinderen te beïnvloeden. Zo slaagden kinderen erin om met behulp van de programmeertaal kennis te construeren.

Een eenduidige definitie voor de term ‘computational thinking’ is er niet omdat deze op verschillende manieren kan benaderd worden (Bastiaensen & De Craemer, 2017). De definiëring van Jeanette Wing (2006) verwierf heel wat bekendheid en luidt als volgt: ‘Computational thinking is het oplossen van problemen, ontwerpen van systemen en het begrijpen van menselijk gedrag door gebruik te maken van concepten uit de computerwetenschappen.’ (p.33). Deze definiëring werd in 2011 door Wing als volgt bijgesteld: ‘Computational thinking is het denkproces waarbij problemen en hun oplossingen zo worden geformuleerd dat ze kunnen worden gepresenteerd in een vorm die effectief kan worden uitgevoerd door een informatie verwerkende tussenpersoon, zoals een computer of een mens, of een combinatie van beide’ (Wing, 2011). De definiëring van SLO (2020a) sluit

hier grotendeels bij aan ‘Computational thinking is een verzameling van denkprocessen waarbij gebruikgemaakt wordt van het (her)formuleren van problemen en het organiseren, analyseren en representeren van gegevens opdat problemen met behulp van ICT-technieken en -gereedschappen kunnen opgelost worden’. Resnick et al. (2009) definiëren computational thinking nog iets ruimer en stellen dat computational thinking je in staat stelt om jezelf uit te drukken met behulp van digitale media. Volgens Lee et al. (2011) kan computational thinking helpen bij het ontwikkelen van het vermogen om te ontdekken, creëren, innoveren en het leert je wat technologie te bieden heeft in deze maatschappij.

Computational thinking kan om bovengenoemde redenen gezien worden als een fundamentele vaardigheid die op vroege leeftijd gestimuleerd moet worden bij kinderen. Computational thinking heeft ook een duidelijke transfer naar andere vakgebieden. Het leren classificeren en categoriseren van informatie kan ook een meerwaarde bieden voor vakken, zoals aardrijkskunde, geschiedenis en biologie. Het ontwikkelen van de onderliggende concepten van computational thinking kan dus ook voor andere schoolvakken voordelen bieden (Guggemos, 2021; Wing, 2006). Bocconi et al. (2016) hebben aan de hand van een vergelijkende literatuurstudie zes concepten van computational thinking gedefinieerd die in vrijwel elk theoretisch raamwerk voorkomen: algoritmisch denken, decompositie, automatisering, abstractie, debuggen en generalisatie (zie Bijlage A voor een overzicht van deze concepten en hun definiëring).

Om deze concepten te concretiseren en te stimuleren bij leerlingen is het leren programmeren een geschikte leeractiviteit. Hierbij kan gebruikgemaakt worden van verschillende soorten programmeeromgevingen (Bastiaensens & De Craemer, 2017; Edwards, 2005; Kennisnet, 2016). Deze kunnen onderverdeeld worden in vier categorieën: tekstueel, visueel, unplugged (programmeren zonder computer) en tangible (Noone & Mooney, 2018). In het onderwijs wordt vaak geopteerd om leerlingen via visuele programmeeromgevingen te

leren programmeren omdat deze vrij toegankelijk zijn voor beginners omwille van de beschikbaarheid (Resnick et al, 2009; Shute, Sun & Asbell-Clarke, 2017). Bij visuele programmeeromgevingen waarbij de output zichtbaar is op een beeldscherm is, in tegenstelling tot tekstuele programmeeromgevingen, geen kennis vereist van een programmeertaal. De syntax is reeds geïmplementeerd in visuele codeblokken die de leerlingen moeten aanklikken en slepen (Bocconi et al., 2016; Kong & Wang, 2019). Dit wordt de 'drag and drop'-methode en zorgt ervoor dat het maken van invoerfouten op basis van de syntax onmogelijk wordt waardoor de drempel voor de leerling verlaagd wordt. Om eerstgenoemde redenen zijn visuele programmeeromgevingen zeer geschikt om leerlingen de onderliggende concepten van computational thinking aan te leren (Kong & Wang, 2020).

Visuele programmeromgevingen kunnen van elkaar verschillen wat betreft de output (Bocconi et al., 2016). Zo zijn bijvoorbeeld Scratch, Alice en Robomind visuele programmeeromgevingen met een beeldschermoutput en Lego WeDo 2.0, Mindstorms en EV programmeromgevingen met een tastbare output. Bij visuele programmeromgevingen met een beeldschermoutput wordt de uitvoer enkel getoond op een scherm van bijvoorbeeld een computer of een tablet (Sapounidis et al., 2015). Het voordeel van programmeeromgevingen met een beeldschermoutput is dat abstracte concepten van computational thinking op het beeldscherm gevisualiseerd worden waardoor de cognitieve belasting van de leerling lager wordt. Op het beeldscherm kan tevens ondersteunende (vb. demonstratievideo's, tutorials) en procedurele informatie (vb. gerichte feedback) worden aangeboden, wat op zijn beurt het opbouwen en automatiseren van schema's bevordert (Bocconi, 2016; van Merriënboer, 2017). Bij programmeeromgevingen met een tastbare output worden fysieke objecten geprogrammeerd aan de hand van een programma (Bocconi et al., 2016). Zo worden de programmeeractiviteiten zeer concreet gemaakt voor leerlingen en worden ze gestimuleerd om het object aan te raken, te onderzoeken en te testen. Dit zou een positief effect hebben op

de verwerving van computational thinking skills en enkele zelfregulerende vaardigheden zoals de betrokkenheid en het reflectief denken van de leerling (Bocconi et al., 2016, Lye & Koh, 2014, Rogers et al., 2002).

1.1.3 Zelfregulatie

Volgens Lee et al. (2008) is er een verband tussen de mate van succes in digitale leeromgevingen en het zelfregulerend vermogen van de lerenden. Zo zouden leerlingen die een lager zelfregulerend vermogen hebben meer moeite hebben bij het aanleren van digitale vaardigheden (Lee et al., 2008). Zelfregulatie wordt gezien als één van de fundamentele vaardigheden in de 21ste eeuwse maatschappij waarin verwacht wordt dat we als individu zelfstandig kunnen handelen en verantwoordelijkheid opnemen (SLO, 2020b; SLO, 2021; Zheng et al., 2018). Onderzoek (Panadero, 2017; Zimmerman 1990) heeft tevens de ontwikkeling van zelfregulerende vaardigheden in verband gebracht met de leerprestaties van leerlingen in het onderwijs.

Zelfregulatie is een actief en constructief proces waarbij de lerende zijn leerdoelen bepaalt en zijn eigen cognitieve en metacognitieve processen reguleert om deze doelen te bereiken (Pintrich, 2000). De lerende neemt met andere woorden de verantwoordelijkheid voor zijn eigen leerproces rekening houdend met de eigen capaciteiten (SLO, 2020b). Volgens Zimmerman (1998; 2008) is zelfregulatie een cyclisch proces dat bestaat uit drie fasen: een preactionele (voorbereiding), actionele (uitvoering) en postactionele fase (evaluatie). Tijdens elke fase moet de lerende rekening houden met het cognitieve, metacognitieve en motivationele component van zelfregulatie (SLO, 2019b). Zelfregulatie is namelijk een samenspel van cognitieve, metacognitieve en motivationele processen die het mogelijk maken dat de lerende zijn capaciteiten kan omzetten naar academische prestaties (Zimmerman, 2008). Bijlage B geeft een overzicht van deze componenten en de bijhorende deelvaardigheden.

Volgens Zheng et al. (2018) vinden heel wat leerlingen het moeilijk om zichzelf te reguleren en hun eigen capaciteiten in te schatten om zodoende vertrouwen op te bouwen om taken tot een goed einde te brengen. Dit kan te wijten zijn aan het gebruik van inefficiënte cognitieve strategieën, een gebrek aan (meta)cognitieve kennis en/of onvoldoende motivatie voor de leertaak, de begeleidende leerrkacht die onvoldoende uitgerust is om het zelfregulerend vermogen te ontwikkelen en de schoolvisie die onvoldoende aandacht heeft voor de ontwikkeling van dit vermogen (Lajoie & Azvedo, 2006). Om deze redenen is het belangrijk dat het zelfregulerend vermogen onder begeleiding gestimuleerd wordt aan de hand van betekenisvolle leeractiviteiten (Zheng et al., 2018).

In recent onderzoek (Duffy & Azvedo, 2015; Lee et al. 2008; Winters et al., 2008) wordt dan ook het verband tussen de ontwikkeling van het zelfregulerend vermogen en visuele programmeermomgevingen vaker gelegd. Zo werd reeds aangetoond dat computergestuurde en online leeromgevingen die vertrekken vanuit reële complexe problemen geschikt zijn om de ontwikkeling van zelfregulerende vaardigheden bij leerlingen te stimuleren. Opdrachten binnen visuele programmeeromgevingen vertrekken vaak vanuit een concrete realistische context en zijn opbouwend in complexiteit (Bocconi et al., 2016; Tsukamoto et al., 2015). Zo is het in vele van deze omgevingen mogelijk om te starten met een level dat aansluit bij de beginsituatie van de lerende en daarop verder te bouwen. Tevens is er de mogelijkheid om te differentiëren op het gebied van de aangeboden begeleiding en kan er zo voldoende ruimte gegeven worden aan de leerling om zelf beslissingen te maken.

1.2 Huidige Studie

Gebaseerd op de bestudeerde literatuur wordt in dit onderzoek nagegaan wat het effect is van leren programmeren in visuele programmeeromgevingen op de ontwikkeling van CT-skills en het zelfregulerend vermogen van leerlingen. Daarnaast zal ook worden onderzocht in

welke mate er een verband is tussen de ontwikkeling van het zelfregulerend vermogen en de ontwikkeling van de CT-skills.

De centrale onderzoeksvraag van dit onderzoek luidt als volgt: *'Wat is het effect van leren programmeren in visuele programmeeromgevingen op de ontwikkeling van computational thinking skills en het zelfregulerend vermogen bij bovenbouwleerlingen van het basisonderwijs?'*

Om deze hoofdvraag te beantwoorden zijn volgende deelvragen geformuleerd:

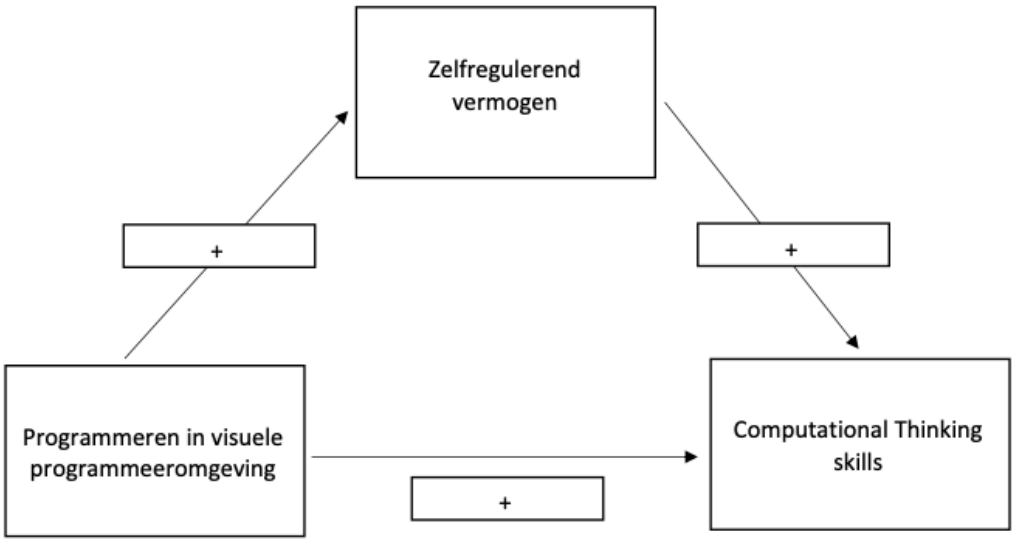
- In welke mate zorgt het leren programmeren met behulp van een visuele programmeeromgeving met een tastbare output voor een groei in de ontwikkeling van CT- skills in vergelijking met het leren programmeren in een programmeeromgeving met een beeldschermoutput?
- In welke mate zorgt het leren programmeren met behulp van een visuele programmeeromgeving met tastbare output voor een groei in de ontwikkeling van het zelfregulerend vermogen in vergelijking met het leren programmeren in een programmeeromgeving met een beeldschermoutput?
- In welke mate is er een verband tussen de ontwikkeling van het zelfregulerend vermogen van de leerling en de ontwikkeling van CT- skills?

Hieruit zijn volgende hypothesen afgeleid:

- De toepassing van visueel programmeren met tastbare output leidt tot een positief meetbaar effect van computational thinking skills in vergelijking met visueel programmeren met een beeldschermoutput.
- Programmeren in een visuele programmeeromgeving met tastbare output leidt tot een hoger niveau van zelfregulerend vermogen in vergelijking met programmeren in een visuele programmeeromgeving met een beeldschermoutput.

- Leerlingen met een hoger zelfregulerend vermogen zullen een grotere groei vertonen in de ontwikkeling van computational thinking skills bij het leren programmeren in een visuele programmeeromgevingen.

Figuur 2 toont het conceptueel model van dit onderzoek met de veronderstelde relaties tussen de variabelen.



Figuur 2. Schematische representatie conceptueel model

Om de veronderstelde relaties uit de vraagstelling te onderzoeken werd in dit onderzoek een kwantitatief quasi-experimenteel onderzoeksdesign gehanteerd (Creswell, 2014). Voor dit experiment werden twee interventiegroepen gevormd waarbij één groep gedurende 6 weken telkens een lesuur per week programmeerles kreeg in een visuele programmeeromgeving met tastbare output en de andere groep in een programmeeromgeving met beeldschermoutput. Dit experiment gebruikte bestaande klasgroepen, enerzijds vanwege praktische overwegingen en anderzijds vanwege het intact houden van de bestaande klasdynamiek (Creswell, 2014). De klasgroepen zullen willekeurig aan één van beide interventiegroepen worden toegewezen.

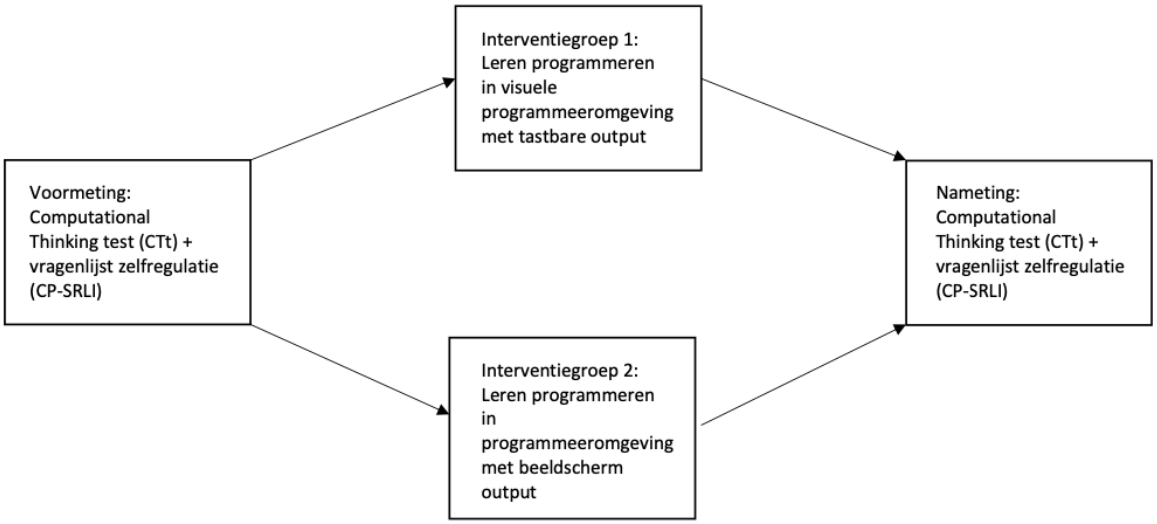
Om het effect van de interventie op de afhankelijke variabelen te meten werd gebruikgemaakt van een pre- en posttest design. De ontwikkeling van de CT-skills en het zelfregulerend vermogen werd zowel voorafgaand als na afloop van de interventie gemeten. In Tabel 1 is het ontwerp van het onderzoek schematisch weergegeven.

Tabel 1

Beschrijving van het onderzoeksdesign

Verdeling over groepen		**Onderzoeksactiviteiten**		
Willekeurige toewijzing per klasgroep	Interventiegroep 1	Pretest: meting CT-vaardigheden en zelfregulatie-vaardigheden	6 programmeerlessen programmeeromgeving met tastbare output	Posttest: meting CT-vaardigheden en zelfregulatie-vaardigheden
Willekeurige toewijzing per klasgroep	Interventiegroep 2	Pretest: meting CT-vaardigheden en zelfregulatie-vaardigheden	6 programmeerlessen programmeeromgeving met visuele output	Posttest: meting CT-vaardigheden en zelfregulatie-vaardigheden

2. Methode



Figuur 3: Flowchart research design

2.1 Deelnemers

De beoogde deelnemers van dit onderzoek zijn bovenbouwleerlingen van het basisonderwijs in Vlaanderen die voor het eerst leren programmeren binnen het schoolcurriculum. De bovenbouw van het basisonderwijs bestaat in het Vlaamse onderwijs uit een 4de, 5de en 6de leerjaar. De leeftijd van de leerlingen varieert hierbij doorgaans van 9 tot 12 jaar. Volgens een poweranalyse met het programma *GPower 3.1* zijn er op basis van t-testen met twee onafhankelijke groepen in totaal 90 deelnemers nodig bij een effectgrootte Cohen's *d* van .06, α-niveau van .05 en een tweezijdige toetsing. Voor het onderzoek werden in totaal 93 leerlingen benaderd. 74 van de 93 benaderde leerlingen kregen toestemming om deel te nemen aan het onderzoek. 44 leerlingen werden toegevoegd aan de interventiegroep die programmerde met een beeldschermoutput, 30 leerlingen aan de interventiegroep die programmeerden met een tastbare output.

2.2 Meetinstrumenten en Materialen

Lego WeDO 2.0

Om de leerlingen te leren programmeren in een visuele programmeeromgeving met tastbare output werd gebruikgemaakt van LEGO WeDo 2.0. WeDo 2.0 is een robotsysteem (zie figuur 4) dat volledig zelf ontworpen, gebouwd en geprogrammeerd kan worden. De leerlingen kunnen de robot programmeren door gebruik te maken van de computerapplicatie WeDo 2.0 waarin ze visuele programmeerblokken moeten slepen en plaatsen (drag and drop, zie figuur 5). Door de blokken in een bepaalde volgorde te plaatsen wordt de robot geprogrammeerd en kan deze bepaalde taken uitvoeren. De robot heeft een bewegingssensor die aan de hand van een bluetoothconnectie verbonden is met de applicatie. De opdrachten van WeDo 2.0 vertrekken steeds vanuit concrete maatschappelijke problemen. De leerlingen worden hierbij uitgedaagd om onderzoeksvragen op te stellen en deelproblemen op te lossen.

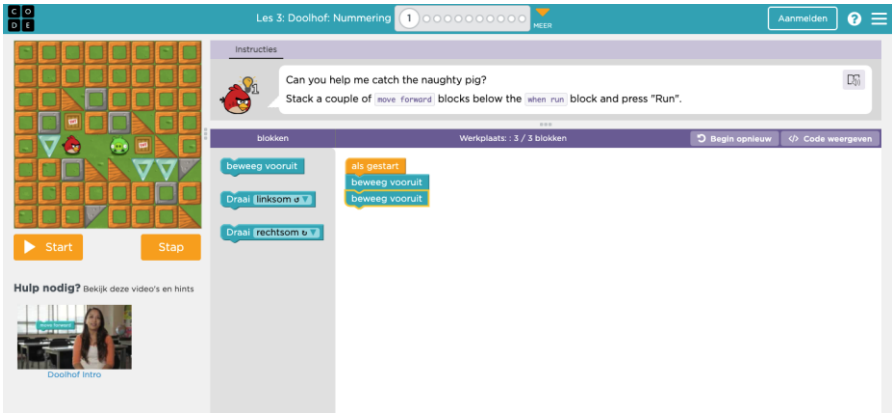
Figuur 4. Voorbeeld programmeerbare robot

Figuur 5. Voorbeeld 'drag and drop'-methode in applicatie

Code.org

De interventiegroep die leerde programmeren in een programmeeromgeving met beeldschermoutput maakte gebruik van het online codeerplatform Code.org. Dit platform wordt wereldwijd gebruikt om kinderen de basisbeginselen van het programmeren aan te leren aan de hand van actviteiten waarbij de leerlingen codeblokken moeten slepen en plaatsen om te programmeren (drag and drop). Concepten die geoefend worden met dit programma zijn: sequenties, loops, voorwaarden, functies (met parameters), variabelen, decompositie, patroonherkenning, abstrahering en algoritmisch denken.

Om deze concepten te leren biedt Code.org aangepaste lessenreeksen aan per leeftijdscategorie waarbij leerlingen per les een nieuw concept aangeleerd krijgen en vervolgens een aantal oefeningen hierop kunnen maken. De leerkracht kan elke nieuwe les starten met een klassikale instructie en demonstratie en in het programma zijn ook videotutorials aanwezig waardoor ook volledig zelfstandig gewerkt kan worden. Dit maakt het mogelijk om te differentiëren, waarbij leerlingen die meer uitdaging nodig hebben zelfstandig aan het werk kunnen en aan een sneller tempo kunnen werken. Het programma maakt het ook mogelijk voor leerlingen om hun eigen leerproces te monitoren. Zo kunnen ze zien welke oefeningen ze reeds gemaakt hebben en welke nog niet. Er is tevens de mogelijkheid om te zien welke concepten van computational thinking reeds worden beheersd en welke nog niet. Als een goed beheersingsniveau wordt bereikt krijgt de leerling hiervoor een trofee. De bedoeling is om alle trofeeën te behalen door te oefenen op concepten die ze nog niet voldoende beheersen.



Figuur 6. Voorbeeld 'drag and drop'-methode in applicatie

Computational Thinking test (CTt)

Om het ontwikkelingsniveau van de computational thinking skills bij de deelnemende leerlingen in de pre- en posttest te meten werd gebruikgemaakt van de gevalideerde Computational Thinking Test (CTt) (Román-González et al., 2017). De test is oorspronkelijk ontworpen voor leerlingen van de eerste graad in het secundair onderwijs en kan ook ingezet worden in de bovenbouw van het basisonderwijs. Voor het huidige onderzoek werd deze test vertaald naar het Nederlands, aangezien gewerkt wordt met leerlingen uit het Nederlandstalig onderwijs waar de kennis van de Engelse taal zeer beperkt is. De vertaalde test werd voorgelegd aan een expertengroep die bestond uit de zorgcoördinator van de betreffende school, een leerkracht voortgezet onderwijs Engels-Nederlands en een leerkracht voortgezet onderwijs informatica/programmeren. Op basis van de feedback van deze experten werd de test aangepast.

De CTt is een meerkeuzetest met 4 antwoordmogelijkheden die bestaat uit 28 items die leerlingen individueel moeten oplossen binnen een tijdsbestek van 45 minuten. Computational thinking wordt op deze test geoperationaliseerd door de volgende concepten: loops, conditionals, sequencing, completion, debugging, basic directions en functions. Een

juist antwoord op een item levert 1 punt op, de score op de test kan hierdoor variëren van 0 tot 28 punten.

Om de betrouwbaarheid van het instrument te achterhalen werd een Cronbach's alpha berekend. Voor de leeftijdscategorie van de leerlingen die aan dit onderzoek zullen deelnemen (9-12j.) werd een Cronbach's alpha berkend van 0.721 (Román-González et al., 2017). Uit de literatuur blijkt dat een Cronbach's alpha van 0.70 overeenstemt met een aanvaardbare betrouwbaarheid (Field, 2018; Román-González et al., 2017).

Zelfregulatietool

Om het zelfregulerend vermogen van leerlingen in kaart te brengen (zowel pre- als posttest) werd de zelfregulatietool gehanteerd. Deze tool is een vragenlijst bestaande uit 75 items en is de Nederlandtalige adaptie van de *Children's Perceived use of Self-regulated Learning Inventory* (CP-SRLI) (Vandevelde et al., 2013). Dit meetinstrument is ontwikkeld voor leerlingen van de bovenbouw van het basisonderwijs en dient individueel ingevuld te worden. De ontwerpers rapporteren dat leerlingen gemiddeld 30 minuten nodig hebben om deze in te vullen.

De vragenlijst bestaat uit 15 schalen die peilen naar concepten van zelfregulatie: 1) taakanalyse: achterhalen wat er wordt gevraagd bij een opdracht (6 items), 2) planning: planningsvaardigheden zoals tijdindelen (4 items), 3) oppervlakkige leerstrategieën: leerstrategieën gericht op kennisreproductie (4 items), 4) diepgaande leerstrategieën: leerstrategieën gericht op het koppelen aan voorgaande kennis en betekenisvol maken van de leerstof (10 items), 5) doorzettingsvermogen: het doorzettingsvermogen en concentratie tijdens opdrachten(6 items), 6) monitoring: nadenken over de manier van werken tijdens een opdracht (7 items), 7) motivatiestrategieën (4 items), 8) zelfevaluatie – productevaluatie: evaluatie van de uitgevoerde opdracht (3 items), 9) zelfevaluatie – procesevaluatie: evaluatie van het proces van de opdracht (4 items), 10) externe regulatie: motivatie voor school omwille

van externe factoren zoals beloningen (3 items), 11) geïntrojecteerde regulatie: motivatie om voor school te werken omwille van verwachtingen ouders of school (4 items), 12) geïdentificeerde regulatie: motivatie om voor school te werken omdat het individu dit belangrijk vindt (4 items), 13) interne regulatie: werken voor school omdat dit plezier en voldoening geeft (3 items), 14) zelfeffectiviteit-zelfregulatie: inschatting van de leerling hoe goe hij zijn leerproces kan reguleren (9 items), 15) zelfeffectiviteit-motivatie: inschatting van de leerling hoe goed hij zich kan motiveren tijdens het leren (4 items). Deze 15 schalen omvatten cognitieve, metacognitieve en motivationele concepten van zelfregulerend leren. Antwoorden op de vragen doen de leerlingen door aan te geven op een 5 puntsschaal hoe goed de stelling bij hen past. Hierbij staat 1 voor helemaal niet akkoord en 5 voor helemaal akkoord. Per schaal wordt een gemiddelde score berekend, hoe hoger de score hoe groter het zelfregulerend vermogen van de leerling (Vandevelde et al., 2013).

Om de betrouwbaarheid van de schalen te achterhalen hebben de ontwerpers het betrouwbaarheidscoëfficiënt Bentler's ρ berekend voor leerlingen van de bovenbouw (10-12j.) in het Vlaamse onderwijs (Vandevelde et al., 2013). Uit deze betrouwbaarheidsanalyse blijkt dat alle schalen een aanvaardbare tot goede betrouwbaarheid hebben ($\rho > 0.60$) uitgezonderd de schaal 'planning' ($\rho = 0.54$). Uit de resultaten van de confirmatorische factor analyse (CFA) blijkt echter dat de items van de schaal wel goed aansluiten bij het model. Desondanks de lagere interne consistentie kan de schaal dus wel gebruikt worden.

2.3 Procedure

Na de formele goedkeuring van de Commissie Ethische Toetsing Onderzoek van de Open Universiteit (cETO) en nadat voldaan werd aan alle vereisten om in België onderzoek te voeren, werd goedkeuring gevraagd aan de directie van een basisschool in West-Vlaanderen om deel te nemen aan het onderzoek. Aan de betrokken leerkrachten werd het opzet, doel en het verloop van het onderzoek voorgesteld. De groepen werden vervolgens in

overeenstemming met de leerkrachten willekeurig in een experimentele of controleconditie ingedeeld. Voor iedere interventiegroep werden zes lesmomenten vastgelegd van 50 minuten waarin de programmeerlessen werden gegeven.

De ouders/voogd van de leerlingen werden aan de hand van een informatiebrief (zie bijlage c) op de hoogte gesteld van het onderzoek. Bij deze brief werd een toestemmingsformulier bijgevoegd voor leerlingen onder de 12 jaar. Van leerlingen die geen toestemming kregen werd geen data meegenomen in het onderzoek. Deze leerlingen namen wel deel aan de programmeerlessen, omdat dit een onderdeel is van het onderwijsprogramma van de school.

Het onderzoek startte met een pretest waarbij de leerlingen de CP-SRLI en de CTt invulden om hun zelfregulerend vermogen en CT- skills in kaart te brengen. Beide testen werden afgelegd in de eigen klasruimte. De testen werden digitaal afgenomen via het schoolplatform wegens het vertrouwde karakter hiervan voor de leerlingen. Voorafgaand aan de interventie werd door de onderzoeker uitleg gegeven over de tools die tijdens het onderzoek gebruikt worden. Voor de groep die werkte met Lego WeDo 2.0 ging dit specifiek over de werking van de applicatie en hoe de robot hiermee geprogrammeerd kan worden. De interventiegroep die werkte met Code.org kreeg uitleg over de website, het programma dat ze moesten doorlopen en het systeem van ‘drag and drop’.

De interventie in beide groepen bestond uit zes sessies van 50 minuten waarbij de leerlingen leerden programmeren in de visuele programmeeromgeving Lego WeDo 2.0 of Code.org. Deze sessies werden gegeven door de klasleerkracht en stonden onder begeleiding van de onderzoeker. De interventiegroep die leerde programmeren met Lego WeDo 2.0 werkte per sessie één project uit waarbij ze de robot een bepaalde handeling moesten laten uitvoeren. Elk project bestond uit drie fasen: onderzoeks-, creatie- en evaluatiefase. In de onderzoeksfase leerden de leerlingen de achtergrondinformatie over wat ze uiteindelijk zullen

moeten creëren aan de hand van video's, in de creatiefase gingen ze zelf aan de slag met het programmeren. In de evaluatiefase deelden ze hun werk met de klasgroep en vulden ze een zelfevaluatie in. In totaal zijn dit zes projecten die terug te vinden zijn in de applicatie waarbij de leeractiviteiten opbouwend zijn in moeilijkheidsgraad en afbouwend in begeleiding. De interventiegroep die programmeerde met Code.org volgde de beginnerscursus die beschikbaar is via de website en bedoeld is voor de leeftijdscategorie van 9-12 jaar. Deze bestond uit 19 lessen die volgende concepten aanleerden: sequenties, loops, voorwaarden, functies, variabelen, decompositie, patroonherkenning, abstrahering en algoritmisch denken. De bedoeling was om per programmeersessie ongeveer 3 lessen aan te reiken. Per les konden de leerlingen 10 oefeningen maken die oplopend waren in moeilijkheid. Leerlingen die sneller de basisbeginselen van het programmeren onder de knie hadden konden op deze manier extra worden uitgedaagd en konden de lessen eveneens zelfstandig doorlopen.

Na de zes programmeersessies werd een nameting gehouden waarbij opnieuw het zelfregulerend vermogen en de CT- skills gemeten werden. Aan het eind van het onderzoek werden de begeleiders gedebrieft.

2.4 Data-Analyse

Om de kwantitatieve data te analyseren werd gebruikgemaakt van het programma IBM SPSS Statistics 28. De data werd vooreerst gecontroleerd op mogelijke outliers, invoerfouten en missende waarden. De deelnemers kregen aan het begin van het onderzoek een respondentnummer toegewezen.

Om het effect van het leren programmeren in de visuele programmeeromgevingen op de ontwikkeling van computational thinkings skills en het zelfregulerend vermogen te controleren werden t-toetsen gehanteerd. Hierbij werd gekeken of er een statistisch significant verschil in CT- skills en zelfregulerend vermogen in de nameting in verglijking met de voormeting. Om het verband tussen de ontwikkeling van CT-skills en het zelfregulerend

vermogen te onderzoeken werd een correlatie-regressieanalyse toegepast. De aannames van normaliteit en homogeniteit werden telkens getest. Indien de data niet voldeed aan de vereiste assumpties werd een bootstrapping-procedure toegepast. Voor dit onderzoek werd een significantieniveau van $p < .05$ gehanteerd.

3. Resultaten

3.1 Visuele Programmeeromgevingen en Ontwikkeling CT-skills

In totaal hebben 74 leerlingen de CT-test afgelegd voor en na de interventiefase. 44 van deze leerlingen leerden programmeren in een visuele programmeeromgeving met een visuele output en 30 leerlingen met een tastbare output. In tabel 2 is een overzicht terug te vinden van de gemiddelden, standaarddeviaties en het mediaan zowel van de voormeting als de nameting.

Tabel 2

Gemiddelden, standaarddeviaties en mediaan van de scores op de CT-test van de pre- en posttest

Variabelen		Voormeting			Nameting		
	n	*M*	*SD*	*Mdn*	*M*	*SD*	*Mdn*
Totaal CT-test (28)							
Totaal	74	11.41	3.11	12.00	13.55	3.61	13.00
Beeldschermoutput	44	12.77	2.67	13.50	15.36	3.10	16.00
Tastbare output	30	9.40	2.62	9.50	10.90	2.51	12.00
CT-skill sequencing (16)							
Totaal	74	6.38	.99	7.00	8.28	2.24	8.00
Beeldschermoutput	44	7.05	1.82	7.00	9.23	2.32	10.00
Tastbare output	30	5.40	1.78	5.00	6.90	1.15	7.00
CT-skill completion (8)							
Totaal	74	3.51	1.66	3.00	3.46	1.51	3.00
Beeldschermoutput	44	4.00	1.46	4.00	4.18	1.13	4.00
Tastbare output	30	2.80	1.69	2.50	2.40	1.38	2.00
CT-skill debugging (4)							
Totaal	74	1.51	1.96	2.00	1.81	.93	2.00
Beeldschermoutput	44	1.73	1.02	2.00	1.95	.83	2.00
Tastbare output	30	1.20	.89	1.00	1.60	1.04	1.50

Noot. CT = computational thinking; Totaal CT-test = aantal items correct; *M* = gemiddelde; *SD* = standaarddeviatie; *Mdn* = mediaan.

We kunnen vaststellen dat de totale groep leerlingen gemiddeld een hogere score behaalde tijdens de nameting in vergelijking met de score van de voormeting. Als we kijken naar de deelvaardigheden voor de totale groep dan zien we dat de CT-skills sequencing en debugging gemiddeld hoger scoren tijdens de nameting. De CT-skill completion scoort gemiddeld lager.

De eerste hypothese van dit onderzoek stelt dat programmeren in een visuele programmeeromgeving met tastbare ouput voor een grotere groei in de onwtikkeling van CT-skills zorgt in vergelijking met een beeldschermoutput. Voor het uitvoeren van de t-testen werden de assumpties van normaliteit en homogeniteit van varianties onderzocht. De data van beide interventiegroepen bleek niet normaal verdeeld te zijn ($p < .05$). Om deze reden werd geopteerd om de t-testen uit voeren met bootstrapping. In tabel 3 is een overzicht terug te vinden van de uitkomsten van de analyses waarbij een vergelijking werd gemaakt tussen de verschillende interventiegroepen en de mate van ontwikkeling van de CT-skills.

Tabel 3

Vergelijking gemiddeldes ontwikkeling CT-skills tussen voor- en nameting interventiegroepen

	BO		TO				
Variabelen	*M*	*SD*	*M*	*SD*	*t*	*p*	*d*
Totaal CT-test	2.59	2.11	1.50	2.19	2.15	.03	.51
CT-skill sequencing	2.18	1.32	1.50	1.83	1.75	.09	.44
CT-skill completion	.18	1.69	-.40	1.83	1.41	.16	.33
CT-skill debugging	.23	1.01	.40	1.45	-.60	.55	-.14

Noot. CT = computational thinking; BO = beeldschermoutput; TO = tastbare output; *M* = gemiddelde; *SD* = standaarddeviatie; *t* = t-waarde; *p* = p-waarde; *d* = effectgrootte

Een interpretatie van deze resultaten laat zien dat er een significant verschil is tussen beide interventiegroepen wat betreft de ontwikkeling van de CT-skills, als we de resultaten van de nameting vergelijken met de voormeting. We kunnen vaststellen dat leerlingen die leerden programmeren in een visuele omgeving met een beeldschermoutput een significant

hogere groei vertonen op de totale score van de CT-test in vergelijking met leerlingen die programmeerden met een tastbare ouput. Als we kijken naar de ontwikkeling van de deelvaardigheden van CT dan stellen we vast de interventiegroep met beeldschermoutput een hogere ontwikkeling vertoont op het gebied van sequencing en completion, deze verschillen zijn echter niet significant. De interventiegroep met tastbare output vertoonde dan weer een grotere groei op de deelvaardigheid debugging, maar ook dit verschil is niet significant.

De hypothese dat leren programmeren met een tastbare output leidt tot een significant hogere groei op de ontwikkeling van CT-skills kan op basis van deze resultaten worden verworpen.

3.2 Visuele Programmeeromgevingen en Ontwikkeling Zelfregulerend Vermogen

De tweede hypothese van dit onderzoek stelt dat leren programmeren met een tastbare output leidt tot een significant hogere groei van het zelfregulerend vermogen van de leerling in vergelijking met het leren programmeren met een visuele output. Tabel 4 geeft een overzicht van de gemiddelde gerapporteerde scores van het zelfregulerend vermogen van de voor- en nameting.

Tabel 4

Vergelijking gemiddeldes zelfregulerend vermogen tussen voor- en nameting interventiegroepen

		Voormeting		Nameting	
Interventiegroep	*n*	*M*	*SD*	*M*	*SD*
Totaal	74	3.28	.37	3.32	.40
Beeldschermoutput	44	3.21	.34	3.32	.43
Tastbare output	30	3.38	.39	3.33	.37

Noot. *M* = gemiddeld zelfregulerend vermogen op een schaal van 1-5 (1=laag, 5 = hoog); *SD* = standaarddeviatie

Als we kijken naar de totale onderzoeksgroep dan stellen we een groei vast van het zelfregulerend vermogen tijdens de nameting in vergelijking met de voormeting. Als we de

twee interventiegroepen bestuderen dan zien we dat de groep die programmeerde met een beeldschermoutput gemiddeld een hogere groei vertoont dan de groep die programmeerde met een tastbare output. Het gerappporteerde zelfregulerend vermogen van de tastbare groep gaat iets achteruit. De verschillen tussen beide interventiegroepen zijn echter niet significant $t(72) = 1.94$, $p = .08$.

De hypothese dat leren programmeren met een tastbare output leidt tot een hogere groei van het zelfregulerend vermogen in vergelijking met leren programmeren met een visuele output kan op basis van deze resultaten worden verworpen.

3.3 Invloed Zelfregulerend Vermogen op Ontwikkeling CT-skills

De derde hypothese van dit onderzoek stelt dat er een verband is tussen de ontwikkeling van het gerapporteerde zelfregulerend vermogen van de leerling en de ontwikkeling van de CT-skills. Er wordt verondersteld dat leerlingen die een hoger zelfregulerend vermogen rapporteren, een grotere groei zullen vertonen van de CT-skills. Om deze hypothese te onderzoeken werd gebruikgemaakt van een lineaire regressieanalyse.

Uit de resultaten van deze analyse blijkt dat er een zwakke correlatie is tussen de ontwikkeling van het zelfregulerend vermogen en de ontwikkeling van de CT-skills, $r\ (73) = .06$, $p = .31$. Vervolgens werd nagegaan of de ontwikkeling van het zelfregulerend vermogen een goede voorspeller zou kunnen zijn voor de ontwikkeling van de CT-skills. Uit de resultaten blijkt het zelfregulerend vermogen geen goede voorspeller te zijn voor de ontwikkeling van de CT-skills, $R^2 = .004$, $F(73) = .256$, $p = .61$.

De hypothese dat leerlingen met een hogere groei in de ontwikkeling van het zelfregulerend vermogen, een grotere groei zouden vertonen in de ontwikkeling van de CT-skills kan worden verworpen.

4. Discussie

De centrale onderzoeksvraag van dit onderzoek luidde als volgt: *'Wat is het effect van leren programmeren in visuele programmeeromgevingen op de ontwikkeling van computational thinking skills en het zelfregulerend vermogen bij bovenbouwleerlingen van het basisonderwijs?'*. Om deze onderzoeksvraag te kunnen beantwoorden werden drie deelvragen en bijhorende hypotheses geformuleerd.

Vooreerst werd de hypothese gesteld dat leren programmeren in een visuele programmeeromgeving met tastbare output zou leiden tot een grotere groei in de ontwikkeling van de CT-skills in vergelijking met het leren programmeren met een visuele output. Uit de resultaten van de data-analyse blijkt dat er aanwijzingen zijn dat deze hypothese niet klopt en dat visuele programmeeromgevingen met een beeldschermoutput leiden tot een betere ontwikkeling van de CT-skills in tegenstelling tot programmeeromgevingen met een tastbare output. Deze bevindingen sluiten aan bij eerder onderzoek naar de voordelen van visuele programmeeromgevingen met een beeldschermoutput (Sapounidis et al., 2015). Het beeldscherm biedt de mogelijkheid om ondersteunende en procedurele informatie aan te bieden aan de lerende en kan zodoende ook gerichte en onmiddellijke feedback geven op acties van de lerende. Dit bevordert de opbouw van schema's en stelt de lerende in staat om snel bij te sturen (Bocconi, 2016; van Merriënboer, 2017). Programmeren met een tastbare output biedt het voordeel dat de lerende heel wat creatieve vrijheid krijgt om te experimenteren, maar dan ook minder task-based is dan de programmeeromgevingen met beeldschermoutput. Deze verschillen in eigenschappen kunnen mogelijks de snellere groei in de ontwikkeling van de CT-skills bij de groep die programmeerde met een beeldschermoutput verklaren.

De tweede hypothese van dit onderzoek stelt dat programmeren in een visuele programmeeromgeving met tastbare output leidt tot een hoger niveau van zelfregulerend

vermogen in vergelijking met programmeren in een visuele programmeeromgeving met een beeldschermoutput. Uit de resultaten van dit onderzoek blijkt dat de groep die programmeerde met een beeldschermoutput een hogere ontwikkeling vertoont van het zelfregulerend vermogen dan de groep die programmeerde met een tastbare output. De verschillen tussen beide interventiegroepen zijn echter niet significant. Deze resultaten kunnen mogelijks opnieuw verklaard worden door de verschillende eigenschappen van de programmeeromgevingen. De programmeeromgeving met beeldschermoutput die werd gebruikt voor dit onderzoek gaf onmiddellijk feedback op de manier van werken tijdens de opdrachten. Er werden ook tips gegeven om efficiënter te werken, waardoor leerlingen hun aanpak snel konden bijsturen. Hoewel het reflectief denken ook gestimuleerd wordt bij een programmeeromgeving met een tastbare output, verloopt dit leerproces mogelijks minder snel omdat de leerlingen eerder zelf moeten uitproberen middels trial-and-error wat de efficiëntste strategieën zijn om tot de oplossing te komen (Bocconi et al., 2016, Lye & Koh, 2014, Rogers et al., 2002). Het is uiteraard mogelijk dat de effecten hiervan slechts op langere termijn zichtbaar zijn en dat de termijn van zes weken van dit onderzoek hiervoor ontoereikend was.

De derde en laatste hypothese stelt dat leerlingen met een hoger zelfregulerend vermogen een grotere groei zullen vertonen in de ontwikkeling van CT- skills bij het leren programmeren in een visuele programmeeromgeving met een visuele output en een visuele programmeeromgeving met een tastbare output. Uit de resultaten blijkt een zwakke, maar geen statistisch significante correlatie tussen de ontwikkeling van het zelfregulerend vermogen en de ontwikkeling van CT-skills. Dit gaat in tegen de verwachtingen vanuit de literatuur (Duffy & Azvedo, 2015; Lee et al. 2008; Winters et al., 2008) dat er een verband is tussen de ontwikkeling van CT-skills en de groei van het zelfregulerend vermogen. In de data zijn er zelfs enkele cases waarbij leerlingen een hogere score behalen op de CTt in de posttest en toch een lager zelfregulerend vermogen rapporteren. Mogelijks kunnen de leeractiviteiten

ervoor gezorgd hebben dat leerlingen een accurater beeld van hun zelfregulerend vermogen kunnen geven en misschien inzien dat hun oorspronkelijk gerapporteerd cijfer uit de pretest te hoog lag.

4.1 Beperkingen van het Onderzoek en Toekomstig Onderzoek

Wegens een aantal beperkingen dienen de resultaten van dit onderzoek met enige voorzichtigheid te worden benaderd. Zo was er de verwachting om 90 leerlingen te kunnen betrekken bij het onderzoek. Vanwege het feit dat niet elke leerling toestemming kreeg is het aantal deelnemers beperkt gebleven tot 74. Dit heeft ook implicaties gehad voor de verdeling van de interventiegroepen. Aangezien gewerkt werd met bestaande klasgroepen en er in bepaalde groepen minder leerlingen toestemming kregen waren de interventiegroepen ongelijk verdeeld. Hierdoor werd meer data verzameld over de groep die programmeerde met een beeldschermoutput. Het werken met bestaande klasgroepen zorgt er ook voor dat bepaalde kenmerken van de groep een invloed kunnen hebben gehad op de resultaten van het onderzoek. Een aanbeveling voor vervolgonderzoeken is dan ook om dit onderzoek te repliceren met grotere onderzoeksgroepen die meer gelijk verdeeld zijn. Het kan ook een mogelijkheid zijn om leerlingen non-random te verdelen bv. naar niveau en/of te testen in hoeverre samenwerken een invloed heeft op de ontwikkeling van deze vaardigheden.

Een tweede beperking kan gevonden worden in de CTt die werd gebruikt om de ontwikkeling van de CT-skills te meten. Deze test heeft de eigenschap om de concepten van CT te meten en is in hoofdzaak task-based. Deze test houdt dus minder rekening met de bredere ontwikkeling van CT, zoals creativiteit en het onderzoekend leren van de leerling. Dit kan mogelijks een bias in de hand hebben gewerkt in het voordeel van de leerlingen die leerden programmeren met een beeldschermoutput. De test die werd gebruikt voor dit onderzoek is echter tot op heden de enige gevalideerde test voor het meten van CT-skills. Inzetten op de ontwikkeling van meetinstrumenten die het bredere karakter van CT, zoals

creativiteit en exploratie, in kaart brengen is dan ook een absolute noodzaak voor verder onderzoek.

Een derde beperking is de beperkte tijdsduur van het onderzoek die mogelijk te kort was om de effecten van beide programmeeromgevingen op lange termijn vast te stellen. Het verschil tussen pre – en posttest bedroeg slechts twee maanden. In deze periode zien we wel al een opmerkelijke vooruitgang van de CT-skills ten opzichte van de pretest, maar het is mogelijk dat deze korte duur in het voordeel speelde van de groep die leerde programmeren met de beeldschermoutput. De effecten van de programmeeromgevingen met een tastbare output zijn omwille van het exploratieve en experimentele karakter mogelijks slechts op lange termijn zichtbaar. Mochten we de leerlingen de kans geven om een volledig schooljaar te werken met beide programmeeromgevingen zouden de resultaten mogelijks anders kunnen zijn. Dit wordt ook bevestigd uit de gesprekken uit de debriefing met de betrokken begeleiders die aangeven dat de groep die programmeerde met tastbare output de laatste lessen een inhaalbeweging aan het maken was en dat ze mogelijks op een effectievere manier enkele strategieën van CT onder de knie kregen. Dit vermoeden kunnen we nu echter aan de hand van deze resultaten niet bevestigen. Een belangrijke aanbeveling kan dan ook zijn om de tijdsduur tussen pre- en posttest te verlengen, idealiter de leerlingen de kans geven om een volledig schooljaar te oefenen.

Een vierde beperking is de zelfregulatietest die werd gebruikt om het zelfregulerend vermogen van de leerlingen in kaart te brengen. Leerlingen moesten hiervoor zichzelf inschatten en het is mogelijk dat dit voor leerlingen van 9-12 jaar nog wat moeilijk was om dit accuraat te doen. We zien bijvoorbeeld bij de posttest dat een aantal leerlingen zichzelf opmerkelijk lager gaan inschatten dan op de pretest. De pretest werd afgenomen aan het begin van het schooljaar waardoor het misschien moeilijker was om juist op elke vraag een

antwoord te geven en dat er misschien ook sprake was van zelfoverschatting. Dit kan een invloed hebben gehad op de resultaten van het onderzoek.

4.2 Maatschappelijke en Wetenschappelijke Relevantie

De maatschappij in de 21ste eeuw is complexer geworden door snelle veranderingen op gebied van informatie en communicatietechnologie (ICT). Om te kunnen functioneren in deze maatschappij is het belangrijk dat leerlingen de 21ste-eeuwse vaardigheden leren beheersen (Saavedra & Opfer, 2012). Allerhande stakeholders (onderwijsinstellingen, werkgevers, maatschappelijke instellingen) verwachten tevens dat jongeren die afstuderen over deze vaardigheden beschikken en houden hier ook rekening mee bij selectieprocedures (Levy & Murnane, 2012). Eén van de taken van ons onderwijs is dan ook om jongeren deze vaardigheden aan te leren aan de hand van verschillende leeractiviteiten doorheen de schoolloopbaan van de jongeren. Dit onderzoek heeft geprobeerd om inzichten te verwerven met betrekking tot de verwerving van CT-skills en zelfregulerende vaardigheden in visuele programmeeromgevingen. Deze inzichten kunnen vervolgens in het werkveld toegepast worden om leerlingen deze vaardigheden aan te leren. Voor leerkrachten zou het een meerwaarde kunnen zijn om te weten of er een verschil is tussen een programmeeromgeving met visuele of tastbare output met betrekking tot het verwerven van deze vaardigheden. Dit kan bijdragen tot het maken van bewustere keuzes voor het inzetten van een bepaalde programmeeromgeving tijdens de lessen. Dit onderzoek heeft kunnen bevestigen dat programmeeromgevingen met een beeldschermoutput een meerwaarde kunnen zijn voor het aanleren van concepten van CT en dat het dus gerechtvaardigd is dat scholen deze gebruiken om leerlingen te laten kennismaken met CT. Vervolgonderzoek is echter aanbevolen om nog accurater het effect van programmeeromgevingen met een tastbare output te kunnen vaststellen.

4.3 Conclusie

Dit onderzoek wees uit dat visuele programmeeromgevingen een prominente rol kunnen spelen in de ontwikkeling van CT, waarbij een output op het scherm kan leiden tot een significante ontwikkeling op CT in vergelijking met omgevingen met een tastbare output. Er werd een zwakke, maar niet significante, correlatie gevonden tussen het zelfregulerend vermogen en de ontwikkeling van CT. Het voorstel is om dit onderzoek te repliceren met grotere en meer gelijk verdeelde groepen. Verder onderzoek is aangeraden naar het ontwikkelen van gevalideerde CT-testen die meerdere benaderingen hebben om een probleem op te lossen, wat ook het creatieve aspect van CT kan stimuleren.

Referenties

Bandura, A. (1986). The explanatory and predictive scope of self-efficacy theory. *Journal of social and clinical psychology*, *4*(3), 359-373. https://doi.org/10.1521/jscp.1986.4.3.359

Bastiaensen, B., & De Craemer, J. (Eds.). (2017). *Zo denkt een computer: programmeren en computationeel denken in het onderwijs*. Departement Onderwijs en Vorming.

Bellanca, J. A. (Ed.). (2010). *21st century skills: Rethinking how students learn*. Solution Tree Press.

Bers, M. U. (2018). Coding and computational thinking in early childhood: the impact of ScratchJr in Europe. *European Journal of STEM Education*, *3*(3), 8. https://doi.org/10.20897/ejsteme/3868

Bjork, R. A., Dunlosky, J., & Kornell, N. (2013). Self-regulated learning: Beliefs, techniques, and illusions. *Annual review of psychology*, *64*, 417-444. https://doi.org/10.1146/annurev-psych-113011-143823

Binkley M. et al. (2012) Defining Twenty-First Century Skills. In: Griffin P., McGaw B., Care E. (eds) *Assessment and Teaching of 21st Century Skills*. (pp. 17-66). Springer. https://doi.org/10.1007/978-94-007-2324-5_2

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education-Implications for policy and practice* (No. JRC104188). Joint Research Centre (Seville site). https://doi.org/10.2791/792158

Boekaerts, M. (1999). Self-regulated learning: Where we are today. *International journal of educational research*, *31*(6), 445-457. https://doi.org/10.1016/S0883-0355(99)00014-2

Care, E., Griffin, P., & McGaw, B. (2012). *Assessment and teaching of 21st century skills*. Springer.

Creswell, J.W. (2014). *Educational research: Planning, conducting and evaluating quantitative and qualitative research.* Pearson Education Limited.

Denning, P. J., & Tedre, M. (2019). *Computational thinking*. Mit Press.

Duffy, M. C., & Azevedo, R. (2015). Motivation matters: Interactions between achievement goals and agent scaffolding for self-regulated learning within an intelligent tutoring system. *Computers in Human Behavior*, *52*, 338-348. https://doi.org/10.1016/j.chb.2015.05.041

Edwards, S. (2005). Identifying the factors that influence computer use in the early childhood classroom. *Australasian Journal of Educational Technology*, *21*(2). https://doi.org/10.14742/ajet.1334

Fanchamps, L. J. A., Slangen, L., Hennissen, P., & Specht, M. M. (2021). The influence of SRA programming on algorithmic thinking and self-efficacy using lego robotics in two types of instruction. *International Journal of Technology and Design Education*, *31*(2), 203-222. https://doi.org/10.1007/s10798-019-09559-9

Field, A. (2018). *Discovering statistics using IBM SPSS statistics*. sage.

Griffin, P., & Care, E. (Eds.). (2014). *Assessment and teaching of 21st century skills: Methods and approach*. Springer.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Guggemos, J. (2021). On the predictors of computational thinking and its growth at the high-school level. *Computers and Education*, *161*https://doi.org/10.1016/j.compedu.2020.104060

Kennisnet (2016). *Computational thinking in het Nederlandse onderwijs*. https://www.kennisnet.nl/app/uploads/kennisnet/publicatie/Computational_Thinking_in_het_Nederlandse_onderwijs.pdf

Kicken, W., Brand-Gruwel, S., van Merriënboer, J., & Slot, W. (2009). Design and evaluation of a development portfolio: How to improve students' self-directed learning skills. *Instructional Science, 37*(5), 453-473. https://doi.org/10.1007/s11251-008-9058-5

Kong, S. C., & Wang, Y. Q. (2019). Assessing programming concepts in the visual block-based programming course for primary school students. In *Proceedings of the 18th European conference on e-learning, ECEL 2019* (pp. 294-302). Academic Conferences and Publishing International. https://doi.org/10.34190/EEL.19.035

Kong, S. C., & Wang, Y. Q. (2020). Formation of computational identity through computational thinking perspectives development in programming learning: A mediation analysis among primary school students. *Computers in Human Behavior, 106*, 106230. https://doi.org/10.1016/j.chb.2019.106230

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, *2*(1), 32-37. https://doi.org/10.1145/1929887.1929902

Lee, T. H., Shen, P. D., & Tsai, C. W. (2008). Applying web-enabled problem-based learning and self-regulated learning to add value to computing education in Taiwan's vocational schools. *Journal of Educational Technology & Society*, *11*(3), 13-25.

Levy, F., & Murnane, R. J. (2012). *The new division of labor*. Princeton University Press.

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. https://doi.org/10.1016/j.chb.2014.09.012

McWhorter, W. I. (2008). *The effectiveness of using LEGO® Mindstorms® robotics activities to influence self-regulated learning in a university introductory computer programming course*. University of North Texas.

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: a systematic review of the literature. *Journal of Computers in Education*, *5*(2), 149-174. https://doi.org/10.1007/s40692-018-0101-5

Panadero, E. (2017). A review of self-regulated learning: Six models and four directions for research. *Frontiers in psychology*, *8*, 422. https://doi.org/10.3389/fpsyg.2017.00422

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Paris, S. G., & Newman, R. S. (1990). Development aspects of self-regulated learning. *Educational Psychologist, 25*(1), 87-102. https://doi.org/10.1207/s15326985ep2501_7

Pintrich, P. R. (2000). The role of goal orientation in self-regulated learning. In *Handbook of self-regulation* (pp. 451-502). Academic Press. https://doi.org/10.1016/B978-012109890-2/50043-3

Pintrich, P. R. (2004). A conceptual framework for assessing motivation and self-regulated learning in college students. *Educational psychology review*, *16*(4), 385-407. https://doi.org/10.1007/s10648-004-0006-x

Prather, J., Becker, B., Craig, M., Denny, P., Loksa, D., & Margulieux, L. (2020). What do we think we think we are doing?: Metacognition and self-regulation in programming. Paper presented at the 2-13. https://doi.org/10.1145/3372782.3406263

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 171-175).

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020). Computational thinking and mathematics using scratch: An experiment with sixth-grade students. *Interactive Learning Environments*, *28*(3), 316-327. https://doi.org/10.1080/10494820.2019.1612448

Rogers, Y., Scaife, M., Gabrielli, S., Smith, H., & Harris, E. (2002). A conceptual framework for mixed reality environments: Designing novel learning activities for young children. *Presence : Teleoperators and Virtual Environment*, *11*(6), 677-686. https://doi.org/10.1162/105474602321050776

Román-González, M., Pérez-González, J., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? criterion validity of the computational thinking test. *Computers in Human Behavior*, *72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Saavedra, A. R., & Opfer, V. D. (2012). Learning 21st-century skills requires 21st-century teaching. *Phi Delta Kappan*, *94*(2), 8-13. https://doi.org/10.1177/003172171209400203

Sapounidis, T., Demetriadis, S., & Stamelos, I. (2015). Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing*, *19*(1), 225-237. https://doi.org/10.1007/s00779-014-0774-3

Schleicher, A. (2012). *Preparing teachers and developing school leaders for the 21st century: Lessons from around the world*. OECD Publishing.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

SLO [21-eeuwse vaardigheden achtergrondinformatie]. (2019a, 4 december). Geraadpleegd op 31 juli 2021, van https://www.slo.nl/thema/meer/21e-eeuwsevaardigheden/achtergrondinformatie/

SLO [Zelfregulering achtergrondinformatie]. (2019b, 4 november). Geraadpleegd op 4 augustus 2021, van https://www.slo.nl/thema/meer/21e-eeuwsevaardigheden/zelfregulering/artikel/

SLO [Computational thinking]. (2020a, 6 januari). Geraadpleegd op 2 augustus 2021, van https://www.slo.nl/vakportalen/vakportaal-digitale-geletterdheid/computational-thinking/

SLO [Zelfregulering]. (2020b, 21 april). Geraadpleegd op 4 augustus 2021, van https://www.slo.nl/thema/meer/21e-eeuwsevaardigheden/zelfregulering/

SLO (2020b). *Leerlijn zelfregulering*. https://www.slo.nl/publish/pages/11324/leerlijn_zelfregulering_maart_2020.pdf

SLO [21-eeuwse vaardigheden]. (2021, 1 juni). Geraadpleegd op 31 juli 2021, van https://www.slo.nl/thema/meer/21e-eeuwsevaardigheden/

Siddiq, F., Gochyyev, P., & Wilson, M. (2017). Learning in digital networks – ICT literacy: A novel assessment of students' 21st century skills. *Computers and Education*, *109*, 11-37. https://doi.org/10.1016/j.compedu.2017.01.014

Thijs, A. M., Fisser, P., & van der Hoeven, M. (2014). *Digitale geletterdheid en 21e eeuwse vaardigheden in het funderend onderwijs: een conceptueel kader*. SLO, Enschede. http://downloads.slo.nl/Repository/21e-eeuwsevaardigheden-conceptueel-kader.pdf

Tsukamoto, H., Takemura, Y., Nagumo, H., Ikeda, I., Monden, A., & Matsumoto, K. (2015). Programming education for primary school children using a textual programming language. Paper presented at the 1-7. https://doi.org/10.1109/FIE.2015.7344187

Turchi, T., & Malizia, A. (2016, september). Fostering computational thinking skills with a tangible blocks programming environment. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 232-233). IEEE.

Van Laar, E., van Deursen, Alexander J. A. M, van Dijk, Jan A. G. M, & de Haan, J. (2020). *Determinants of 21st-century skills and 21st-century digital skills for workers: A systematic literature review*. SAGE Publications. https://doi.org/10.1177/2158244019900176

van Merriënboer, J. J.. (2017, oktober). *Ondersteunen van zelfregulatie*. Presentatie tijdens conferentie zelfregulatie – afsluiting NRO-project, Heerlen.

Vandevelde, S., Van Keer, H., & Rosseel, Y. (2013). Measuring the complexity of upper primary school children's self-regulated learning: A multi-component approach. *Contemporary Educational Psychology*, *38*(4), 407-425. https://doi.org/10.1016/j.cedpsych.2013.09.002

Voogt, J., Brand-Gruwel, S., & Van Strien, J. (2017). Effecten van programmeeronderwijs op computational thinking: een reviewstudie. Geraadpleegd van https://www.nro.nl/wpcontent/uploads/2017/05/003-Antwoord-Rapport-Programmeeronderwijs.pdf

Voogt, J., & Roblin, N. P. (2010). 21st century skills. *Discussienota. Zoetermeer: The Netherlands: Kennisnet*, *23*(03), 2000.

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

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical, and Engineering Sciences*, *366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. (2011). *Research Notebook: Computational Thinking - What and Why?*. Carnegie Mellon.

Winne, P. H. (2001). Self-regulated learning viewed from models of information processing. *Self-regulated learning and academic achievement: Theoretical perspectives*, *2*, 153-189.

Winters, F. I., Greene, J. A., & Costich, C. M. (2008). Self-regulation of learning within computer-based learning environments: A critical analysis. *Educational psychology review*, *20*(4), 429-444. https://doi.org/10.1007/s10648-008-9080-9

Wu, S., & Su, Y. (2021). Visual programming environments and computational thinking performance of fifth- and sixth-grade students. *Journal of Educational Computing Research*, https://doi.org/10.1177/0735633120988807

Zheng, L., Li, X., & Chen, F. (2018) Effects of a mobile self-regulated learning approach on students' learning achievements and self-regulated learning skills. *Innovations in Education and Teaching International*, *55*(6), 616-624. https://doi.org/10.1080/14703297.2016.1259080

Zimmerman, B. J. (1990). Self-regulated learning and academic achievement: An overview. *Educational psychologist*, *25*(1), 3-17. https://doi.org/10.1207/s15326985ep2501_2

Zimmerman, B. J. (1998). Developing self-fulfilling cycles of academic regulation: An analysis of exemplary instructional models. In D. H. Schunk & B. J. Zimmerman (Eds.), *Self-regulated learning: From teaching to self-reflective practice* (pp. 1–19). Guilford Publications.

Zimmerman, B.J., (2008). Investigating self-regulation and motivation: Historical background, methodological developments, and future prospects. *American educational research journal*, *45*(1), pp.166-183. https://doi.org/10.3102/0002831207312909

Zimmerman, B. J., & Schunk, D. H. (Eds.). (2001). *Self-regulated learning and academic achievement: Theoretical perspectives*. Routledge.

Bijlagen

Bijlage A: Basisconcepten computational thinking

Tabel A

Basisconcepten computational thinking

Concept	Definiëring
Abstractie	Het focussen op de belangrijke informatie, negeren van irrelevante details.
Algoritmisch denken	Een geheel van ondubbelzinnige instructies die stapsgewijs uitgevoerd moeten worden om een probleem op te lossen.
Automatisering	Het proces waarbij een reeks van repetitieve taken snel en efficiënt worden uitgevoerd.
Debuggen	Het systematisch toepassen van analyse en evaluatie om uitkomsten te verifiëren en voorspellen.
Decompositie	Het verdelen van een ingewikkeld probleem in verschillende kleinere problemen die afzonderlijk opgelost kunnen worden.
Generalisatie	Een manier om snel nieuwe problemen op te lossen door het zoeken naar gelijkenissen of overeenkomsten tussen en in problemen.

Noot. Overgenomen uit ‘Developing Computational Thinking in Compulsory Education’ van Bocconi et al. (2016) p.18.

Bijlage B: Componenten van zelfregulatie

Tabel B

Componenten van zelfregulatie

Component	Definiëring	Voorbeelden
Cognitie	De cognitieve component omvat de vaardigheden die nodig zijn bij het coderen, opslaan en ophalen van informatie. (Zimmerman, 1990; Boekaerts, 1999; SLO, 2019b; Winne, 2001)	– Het inzetten van leerstrategieën die het leren optimaliseert. – Op systematische wijze informatie verwerven en gebruiken. – Op systematische wijze gebruikmaken van informatiebronnen.
Metacognitie	Het metacognitieve component omvat de vaardigheden die nodig zijn bij het monitoren en begrijpen van cognitieve processen (Boekaerts, 1999; Pintrich, 2000).	– Doelen opstellen – Analyseren van probleemstelling – Opstellen van planning – Organiseren van het leerproces in functie van leerdoelen – Zelfevaluatie en reflectie
Motivatie	Het motivationele component omvat de overtuigingen en en standpunten die het gebruik en de ontwikkeling van de (meta)cognitieve vaardigheden beïnvloeden (Pintrich, 2004; Zimmerman, 2000)	– Zelfeffectiviteit (percepties over het eigen vermogen om de opgestelde doelen te bereiken) (Bandura, 1986) – Doorzettingsvermogen

	– Attribueren (resultaten toeschrijven aan juiste oorzaak)

Bijlage C: Informatiebrief

Beste ouder/voogd,

Middels deze brief willen wij u informeren over een wetenschappelijk onderzoek dat met goedkeuring van de directie gaat plaatsvinden op de school van uw kind.

Voor de deelname van uw zoon/dochter aan dit wetenschappelijk onderzoek is uw schriftelijke toestemming vereist. In deze brief staat uitvoerig beschreven wat dit onderzoek inhoudt en wat er van uw kind wordt verwacht. Mocht u hierover nog vragen hebben, dan kunt u deze stellen aan de onderzoekers (zie gegevens onderaan deze brief).

Inleiding

Snelle veranderingen op het gebied van Informatie en Communicatie Technologie (ICT) zorgen ervoor dat onze maatschappij in de 21ste eeuw steeds complexer wordt. Het gevolg hiervan is dat ook in het onderwijs voldoende aandacht moet besteed worden aan
de ontwikkeling van bepaalde vaardigheden zodat leerlingen kunnen uitgroeien

tot volwaardige deelnemers van de 21ste eeuwse samenleving. Eén van deze
belangrijke vaardigheden is computationeel denken. Om deze vaardigheid te ontwikkelen worden reeds op de school van uw kind programmeerlessen aangeboden.

Doel van het onderzoek

Het doel van dit onderzoek is om op wetenschappelijke wijze na te gaan in welke mate het leren programmeren met visuele programmeeromgevingen een effect heeft op de ontwikkeling van computationeel denken en of het zelfregulerend vermogen (dit zijn vaardigheden als taakanalyse, planning, zelfevaluatie en motivatie) van de leerling hierop

een invloed heeft.
Wat wordt er van de kinderen verwacht?

In klas zal uw kind gevraagd worden om een test te in te vullen met een paar oefeningen waardoor het computationeel denken en het zelfregulerend vermogen gemeten wordt.

Na het volgen van zes programmeerlessen in de klas zal uw kind opnieuw gevraagd worden om de test in te vullen.

De programmeerlessen zijn een vast onderdeel van het leerplan, dus ook bij een eventuele niet-deelname moeten deze lessen door elke leerling gevolgd worden.

Indien u beslist om uw kind niet te laten deelnemen zullen de gegevens van de testen en vragenlijsten niet
meegenomen worden in het onderzoek.

Welke gegevens worden verzameld en waar worden deze voor gebruikt?
- Persoonsgegevens leerling: voornaam en achternaam, leerjaar van uw kind.

- Onderzoeksgegevens: de mate van computationeel denken voor en na de programmeerlessen, de mate van zelfregulering voor en na de programmeerlessen.

De persoonsgegevens worden apart van de onderzoeksgegevens bewaard. De andere gegevens worden gebruikt voor het beantwoorden van de onderzoeksvragen.

Deelname aan het onderzoek

Alleen kinderen uit het 4de, 5de en 6de leerjaar kunnen deelnemen aan het onderzoek. U beslist zelf of uw kind deelneemt aan het onderzoek. Deelname is vrijwillig. Als uw kind deelneemt aan het onderzoek, kunt u zich altijd bedenken en toch stoppen, ook tijdens het onderzoek. U hoeft daarvoor geen reden te geven.

U dient toestemming te geven voor de deelname van uw kind aan dit wetenschappelijk onderzoek, hiervoor moet u online toestemming geven via het Google Forms.

Ethische toetsing en privacy

Dit onderzoek is goedgekeurd door de cETO, de ethische commissie van de Open Universiteit, en voldoet aan alle ethische standaarden die gelden voor mensgebonden onderzoek. **Alle gegevens zullen strikt anoniem en vertrouwelijk worden behandeld. Data zal gecodeerd (niet tot de persoon herleidbaar) worden opgeslagen.** Individuele resultaten worden niet aan derden doorgespeeld, dus bijvoorbeeld de school of een docent krijgt geen inzage in individuele resultaten. U kunt de toestemming voor verwerking van de gegevens van uw kind voor dit onderzoek altijd weer intrekken. De reeds verzamelde gegevens zullen dan nog wel worden gebruikt in het onderzoek. Indien u wilt dat al uw gegevens worden verwijderd, dan kunt u daarom verzoeken. Voor meer informatie over de persoonsgegevensdisclaimer van de OU kunt u terecht op: www.ou.nl/privacy.

Het einde van het onderzoek

Het hele onderzoek is afgelopen als de afnameperiode van de testen en vragenlijsten verstreken is en de onderzoekers de gegevens verwerkt hebben. Indien gewenst informeert de onderzoeker u over de belangrijkste uitkomsten van het onderzoek. Uiteraard zullen de betrokken scholen ook op de hoogte gesteld worden van de belangrijkste uitkomsten van het onderzoek.

Bewaren van de data

De data zal gedurende de looptijd van het onderzoek bewaard worden op een beschermde server van de Open Universiteit en daarna zal de gecodeerde dataset (dus niet tot de persoon te herleiden), voor de wettelijke termijn van 10 jaar, bewaard worden in een online afgeschermde opslagplaats.

Mocht u na het lezen van deze brief nog vragen hebben of wil u extra informatie, dan kunt u altijd (voor, tijdens en na het onderzoek) contact opnemen met het onderzoeksteam.

Met vriendelijke groet,

Mr. M. Vansteenkiste

Dr. N. Fanchamps (hoofdonderzoeker)