

Zelf programmeren

Een toets voor het
informaticaonderwijs

Kwaliteit

Onderzoek

Aantrekkelijker

Motivatie

Wat werkt wel?

Efficiënter onderwijs

Toegevoegde waarde

Effectief

Wat werkt niet?

Onderzoeksreeks ict in het onderwijs

Voorwoord

Dit is de zesendertigste publicatie in de Kennisnet Onderzoeksreeks *Ict in het onderwijs*.

Deze publicatie gaat over het schoolvak informatica, in het bijzonder het leren schrijven van computerprogramma's. Het is een van de eerste publicaties in de reeks waarbij leren omgaan met ict het doel van onderwijs is, in plaats van een middel.

Ernst Koldenhof, Johan Jeuring en Sandra Ruth van de Universiteit Utrecht beschrijven in dit boekje hoe de informaticalessen worden gegeven en wat leerlingen ervan leren.

Het vak informatica is nog jong en het blijkt nog duidelijk een pioniersvak. Leerlingen doen in de lessen maar weinig ervaring op met zelfstandig programmeren en worden onvoldoende voorbereid voor een ict-vervolgopleiding in het hoger onderwijs. Eindtermen zijn vaag, in de methodes komen vooral de lagere leerniveaus (onthouden, begrijpen, direct toepassen) aan bod en de invulling van het vak hangt vaak af van de persoonlijke voorkeuren van de (vaak niet in informatica geschoolde) leraar.

Ict speelt een belangrijke rol om te voorzien in innovatieve producten en diensten. Voor de arbeidsmarkt betekent dit een grote vraag naar hoogwaardig opgeleid personeel en voor het onderwijs de uitdaging om vakmensen met kennis over ict aan te leveren.

Dit onderzoek legt een aantal concrete problemen bloot rondom de voorbereiding van jongeren op deze ict-gerelateerde beroepen. Deze inzichten bieden aanknopingspunten die helpen om een nieuwe fase in het informaticaonderwijs inhoud en richting te geven, voorbij de pioniersfase.

Alfons ten Brummelhuis
Hoofd Onderzoek Kennisnet

Inhoud

1	Informatica- en programmeeronderwijs	4
1.1	Informatica als schoolvak	4
1.2	De leraren informatica	5
1.3	Problemen van het informaticaonderwijs	5
1.4	Een verkennend onderzoek naar het programmeeronderwijs	5
2	Wat hebben de lesmethodes te bieden?	7
2.1	Leerdoelen van het programmeeronderwijs	7
2.2	Voorbeeld van een methode: Enigma	8
2.3	Het raamwerk: een taxonomie van leerniveaus	9
2.4	Welke leerniveaus komen er in de methodes aan bod?	11
2.5	Waaraan opdrachten verder moeten voldoen	12
3	Wat leraren en leerlingen van programmeren vinden	14
3.1	Vier scholen	14
3.2	De werkwijze van de leraren	14
3.3	De mening van de leerlingen	15
4	De leeropbrengst van het programmeeronderwijs	17
4.1	De toetsen	17
4.2	De relatie tussen de methodes en de toetsresultaten	19
5	Conclusie	20
6	Meer weten?	21
6.1	Literatuur	21
6.2	Over het onderzoek	22
6.3	Over de auteurs	22
6.4	Een vraag stellen	22
	Colofon	23

1 Informatica- en programmeeronderwijs

1.1 Informatica als schoolvak

Informatica is een jong vak. Het zit pas sinds 1998 in de bovenbouw van vwo en havo, en dan alleen nog maar als keuzevak. Scholen zijn niet verplicht het aan te bieden en niet meer dan 60% van de scholen heeft het op het programma staan. Het leidt ook niet op voor een informaticastudie in het wo of een ict-opleiding in het hbo: het is een algemeen vormend vak.

In 2007, in de vernieuwde Tweede Fase, kreeg informatica meer lesuren. Voor het profiel Natuur en Techniek werd het een profielkeuzevak en voor de andere drie profielen werd het een keuzevak in de vrije ruimte. Er is geen centraal schriftelijk examen, alleen een schoolexamen. Het totale aantal lesuren in het vwo is 440, in de havo 320 (ongeveer 3 uur per week).

Toen het examenprogramma werd vastgesteld, is gekozen voor breedte. Alle fasen van het traject waarin een computerprogramma ontwikkeld moet worden, komen in het schoolvak informatica aan de orde. Naast de technische kant, het programmeren, gaat het daarbij ook om de mensen die de computers gebruiken. Hoe werken zij – of willen zij werken – met informatie- en communicatiesystemen? De vereiste kennis is gespreid over vier domeinen:

- A *Informatica in perspectief* omvat de geschiedenis van de informatica, de rol van ict in de maatschappij, de functies waarin ict'ers in het algemeen werken en het karakteristieke van hun werk, met name het projectverband.
- B *Basisbegrippen en vaardigheden* gaat over hoe je gegevens digitaal codeert, hoe de hardware en de randapparatuur werkt, welke datatypen en programmastructuren er zijn, hoe je een eenvoudig programma ontwikkelt en hoe een bedrijf in elkaar zit.
- C *Systemen en hun structurering* beschrijft de communicatie in een netwerk, het besturingssysteem, de systeemontwikkeling, de interactie tussen mens en machine en het systeemontwikkeltraject: het testen van een prototype en de vraag of het systeem aan de wensen voldoet.
- D *Toepassingen in samenhang* gaat over projectmanagement.

1.2 De leraren informatica

De leraren informatica hebben in het algemeen geen informatica gestudeerd. Zij hebben een eerstegraads bevoegdheid in een ander vak en zijn opgeleid via het CODI. Dit Consortium Omscholing Docenten Informatica heeft van 1998 (toen informatica examenvak werd) tot 2006 (toen er een eerstegraads opleiding kwam) de leraren informatica opgeleid.

De achtergrond van deze leraren kan variëren van Frans tot wiskunde, van gymnastiek tot economie. Wat zij gemeen hebben was belangstelling voor informatica (geen enkele cursist was blanco), maar het is te verwachten dat hun uiteenlopende achtergrond zal leiden tot zeer diverse opvattingen over het vak. De docenten krijgen ook nog geen formele bevoegdheid voor Natuur, Leven en Technologie (NLT).

Sinds 2006 bestaat er een eerstegraads lerarenopleiding informatica. Dit is een mastersopleiding, wat de mogelijkheid opent om op het gebied van de informaticadidactiek onderzoek te verrichten – iets waar zowel in Nederland als internationaal grote behoefte aan is. Overigens is het aantal studenten tot nu toe beperkt – waarbij ongetwijfeld van invloed is dat in andere ict-sectoren het beroepsperspectief veel aantrekkelijker is.

1.3 Problemen van het informaticaonderwijs

Het schoolvak informatica kampt met diverse problemen. We noemen er een paar, in willekeurige volgorde:

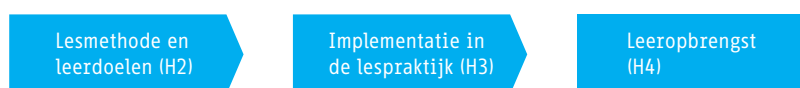
- Er staan zoveel onderwerpen op het programma, dat het gevaar bestaat dat ze allemaal oppervlakkig behandeld worden.
- Er is een discrepantie tussen het leerplan en wat de leraren doen: op de havo doen ze minder dan wat het programma voorschrijft, op het vwo juist iets meer.
- De leraren zijn niet allemaal in dezelfde mate geschoold.
- Een onderdeel als programmeren is voor veel leerlingen een struikelblok (Proulx, 2000).
- Er zijn *stoppers* en *movers*, afhakers en doordouwers (Perkins, 1986). Het is de vraag of het informaticaonderwijs de afhakers voldoende ondersteunt, en anderzijds de doordouwers voldoende uitdaagt.

1.4 Een verkennend onderzoek naar het programmeeronderwijs

Sinds 2007, toen informatica meer lesuren kreeg, is programmeren een belangrijk onderdeel van het vak (Nationaal expertisecentrum leerplanontwikkeling [SLO] 2009). Het is een technisch onderdeel, waarvoor je – in relatief korte tijd – een programmeertaal onder de knie moet zien te krijgen.

Zelfs studenten informatica vinden het moeilijk om te leren programmeren; ze lopen er studievertraging door op en sommigen stoppen zelfs helemaal met hun studie. Hoe komt dat? Waarom is het zo moeilijk om te leren programmeren?

Om een antwoord te vinden op deze vragen bekijken we in dit boekje hoe het programmeeronderwijs wordt gegeven en wat de leeropbrengst ervan is. Daarvoor bekijken we de lesmethodes en de centrale leerdoelen, maar ook hoe de leraren daar in de praktijk mee werken en wat leraren en leerlingen daarvan vinden. Bovendien toetsen we wat de leerlingen hebben geleerd. Schematisch ziet dat er zo uit:



Bij het onderzoek hebben we ons geconcentreerd op bestaande lesmethodes, en wel op twee die veel gebruikt worden: *Enigma* en *Fundament* (zie voor meer informatie: Stichting Enigma, 2007; Van der Laan, 1999). Vier leraren op vier verschillende scholen waren bereid mee te werken aan het onderzoek. Twee van hen werken met Enigma, twee met Fundament. Het is duidelijk dat dit een verkennend onderzoek is: met zo'n bescheiden steekproef kun je geen algemene uitspraken doen, en dus alleen maar voorlopige resultaten boeken.

Beknorte opzet van het onderzoek

Onderzoeksobject: twee leermethodes, Fundament en Enigma, op vier scholen

- Attitudes van de leerlingen: gemeten met attitudetesten voor en na de lessen programmeren (kwantitatief)
- Het programmeeronderwijs:
 1. Lesmethode en leerdoelen: inhoudelijk onderzocht (literatuurstudie)
 2. Perceptie van leraren en leerlingen op het gevolgde programmeeronderwijs: lesobservaties (91 leerlingen) en interviews (kwalitatief)
 3. Leeropbrengst van het programmeeronderwijs: tussentijdse toets en eindtoets van de vakinhoudelijke vorderingen (kwantitatief)

De onderzoeksrapportage (Koldenhof, 2011) met meer gegevens over de uitvoering en opzet van het onderzoek is te raadplegen op de onderzoekswebsite van Kennisnet: onderzoek.kennisnet.nl.

2 Wat hebben de lesmethodes te bieden?

Om iets te kunnen zeggen over de methodes *Enigma* en *Fundament* is het belangrijk om eerst te bekijken welke leerdoelen voor programmeeronderwijs landelijk zijn vastgesteld in de examenprogramma's informatica voor havo en vwo. Sluiten de theorie en de opdrachten in de methodes hierbij aan?

2.1 Leerdoelen van het programmeeronderwijs

In de eindtermen vinden we over programmeren heel weinig. Onder het domein 'Basisbegrippen en vaardigheden' valt het subdomein 'Software', met als tekst:

De kandidaat beheerst eenvoudige datatypen, programmastructuren en programmeertechnieken.

Dat is alles. In 2007 verscheen de *Handreiking schoolexamen informatica havo/vwo*, geschreven door Victor Schmidt. Daaruit kunnen we vier concretere hoofddoelen destilleren (p. 20-21):

(1) Kernbegrippen

Van het objectgericht programmeren moeten de leerlingen de volgende begrippen beheersen:

- object
- objectklasse
- attributen en methodes

Een korte uitleg van deze begrippen is hier op zijn plaats. De 'objecten' zijn onder andere de onderdelen die op het scherm te zien zijn, waarmee de gebruiker informatie kan doorgeven aan de computer. Bijvoorbeeld knoppen en invulvelden. Deze schermobjecten kennen 'attributen' zoals plaats, kleur of opschrift, en 'methodes' die een input van de gebruiker kunnen afhandelen. Er bestaan ook 'logische objecten'; die staan alleen in het geheugen van de computer en zijn niet op het scherm te zien. Een 'objectklasse' is een verzameling objecten met eenzelfde sjabloon. Simpel gezegd kun je een object vergelijken met een stoel, de objectklasse is dan meubels en het attribuut is eikenhout.

(2) Toepassen van kernbegrippen bij het maken van een visueel programma

Leerlingen kunnen deze begrippen toepassen bij de ontwikkeling van een objectgeoriënteerd of visueel programma. Een visueel programma heeft een grafische *user interface* met schermobjecten als knoppen, invulvelden, selec-

tievelen, enzovoorts. In het kernprogramma is het voldoende dat leerlingen het objectbegrip kunnen verbinden met schermobjecten en een visueel programma kunnen schrijven. Dit verbinden gebeurt via programma's, geschreven in een programmeertaal. Je kunt een programma vergelijken met een tekst die bestaat uit letters, woorden en zinnen van een vreemde taal. Leerlingen moeten leren programma's te herkennen, te lezen, de fouten eruit te halen en ze uiteindelijk zelf te schrijven. Als ze dat allemaal kunnen, kunnen ze programmeren.

(3) Een verdiepingsprogramma

Voor vwo-leerlingen en voor havo-leerlingen die informatica willen gaan studeren. In het *verdiepingsprogramma* vwo leren de leerlingen bovendien:

- een programma te schrijven met logische objecten (objecten die enkel in het geheugen van de computer staan en niet als schermobject zichtbaar zijn)
- collectieobjecten in het programma te gebruiken. Collectieobjecten zijn objecten die bestaan uit een verzameling van andere objecten, zoals de *array* en de *array list* (Java)

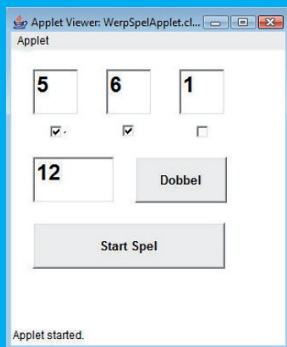
Daarnaast leren vwo-leerlingen 'netjes' te programmeren, dat wil zeggen dat ze gestructureerd leren programmeren. Ook voor havo-leerlingen die opteren voor de hbo-opleidingen Informatica en Technische Informatica is dit een nuttige verdieping.

(4) Praktische toepassingen

Ten slotte biedt het project in de verdiepingsprogramma's de gelegenheid het bovenstaande in een praktische context toe te passen.

2.2 Voorbeeld van een methode: Enigma

Enigma maakt gebruik van de programmeertaal Java. De methode begint met een introductie op de begrippen 'object' en 'klasse'. De leerling krijgt op een computerscherm afbeeldingen te zien met daarop verschillende objecten: knoppen, tekstvelden en selectievakjes. In de tekst wordt uitgelegd wat in Java de namen van de bijbehorende objectklassen zijn: Button, Textfield en Checkbox. Voor de dobbelstenen kent Java geen objectklasse. De leerling moet in de bijbehorende opdracht uit het verwerkingsboek zelf de klasse Dobbelsteen aanmaken. Hij moet dus leren hoe hij een programma moet schrijven om een dobbelsteen te maken.



OPDRACHT

Om ervaring op te doen met het ontwerpen en schrijven van een klasse ga je een eenvoudig dobbelspel programmeren.

In dit spel mag een speler drie keer gooien met maximaal drie dobbelstenen. Hij kan besluiten om een of meerdere stenen te laten liggen. Het doel van het spel is om na drie worpen een zo hoog mogelijke score te behalen.

De interface van het spel staat hiernaast afgebeeld.

Welke objecten zijn er te onderscheiden in dit spel?

Wanneer je naar de interface kijkt dan zie je twee knoppen: Start Spel en Dobbel. Er zijn drie tekstvelden die het resultaat van de worp laten zien en een vierde tekstveld dat de totale score weergeeft.

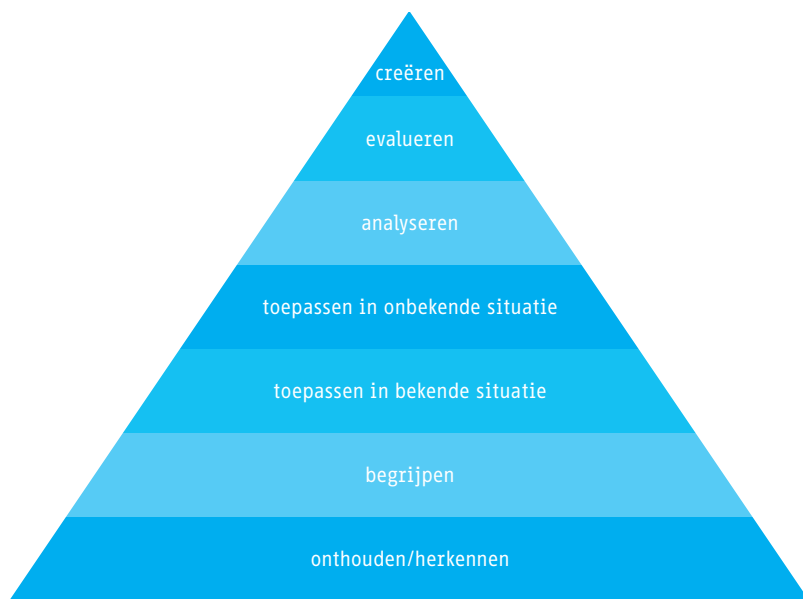
Tenslotte zijn er nog drie selectievakjes. Door een selectievakje aan te vinken geeft de speler aan dat hij deze dobbelsteen laat liggen. Deze objecten zijn instanties van respectievelijk de klassen Button, Textfield en Checkbox.

Behalve de objecten die je op de interface ziet, zijn er nog drie objecten: de dobbelstenen waarmee gegooid moet worden. Java kent geen standaardklasse Dobbelssteen. Die zul je zelf moeten ontwerpen.

Figuur 1: Voorbeeldopdracht uit Enigma. Leerlingen leren hiermee hun begripkennis toe te passen door een eenvoudig dobbelspel te ontwerpen (Barendsen, 2007).

2.3 Het raamwerk: een taxonomie van leerniveaus

Om zicht te krijgen op de lesmethodes *Enigma* en *Fundament* is het wenselijk dat we de opdrachten daarin eerst rubriceren. Daarbij maken we gebruik van de taxonomie van Bloom (1956). In de jaren vijftig van de vorige eeuw onderscheidde de onderwijspsycholoog Benjamin Bloom in de menselijke kennis zes verschillende niveaus. Deze leerniveaus vormen een piramide, die je kunt zien als stappen in het leerproces: hoe hoger in de piramide, hoe beter de leerling de stof beheerst. Om tot een hogere trap te komen moet je de onderliggende trap beheersen. Om bepaalde informatie te begrijpen (trap 2), moet je de informatie eerst kennen (onthouden/herkennen, trap 1), en voor het toepassen (trap 3) is eerst begrip nodig. Als je voor leerlingen een vraag of opdracht maakt, is het belangrijk dat je je afvraagt wat het doel is van de opdracht. Wat wil je dat ze weten of kunnen?



Figuur 2: Hiërarchie van leerniveaus (gebaseerd op Bloom, 1956, zie ook http://www.odu.edu/educ/roverbau/Bloom/blooms_taxonomy.htm)

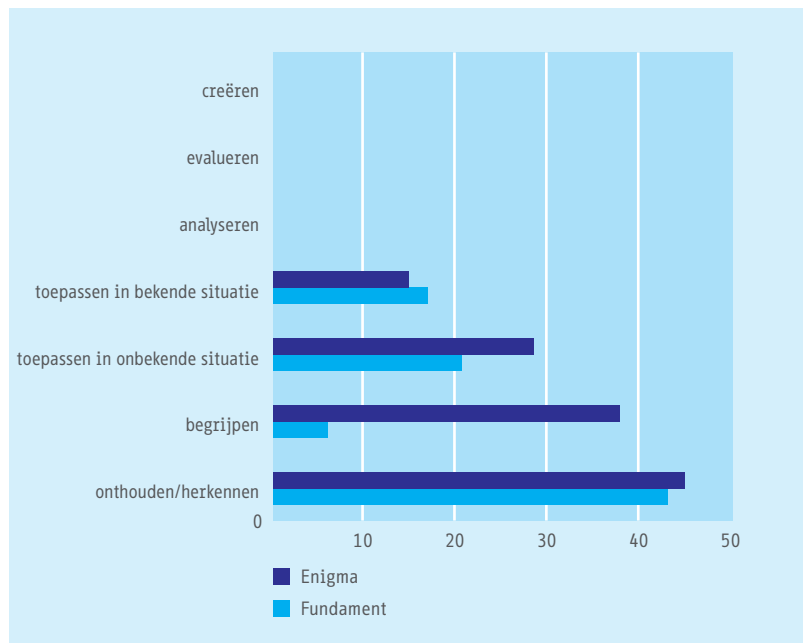
Met deze niveau-indeling hebben we de opdrachten uit de lesmethodes *Enigma* en *Fundament* ingedeeld. Het niveau 'toepassen' van Bloom hebben we in tweeën gesplitst, omdat er toch een wezenlijk verschil is tussen het toepassen van kennis in een bekende situatie en in een onbekende situatie. We onderscheiden dus zeven niveaus:

1. Onthouden en herkennen: de antwoorden direct uit de lestekst kunnen halen
2. Begrijpen: begrip van het onderwerp nodig om het antwoord te kunnen formuleren
3. Toepassen (in bekende situatie): kennis van het onderwerp kunnen toepassen in een situatie die lijkt op een situatie uit de lesstof
4. Toepassen (in onbekende situatie): kennis van het onderwerp kunnen toepassen in een minder bekende situatie
5. Analyseren: verschillende kennisonderdelen met elkaar kunnen vergelijken
6. Evalueren: standpunten over het onderwerp kunnen verdedigen
7. Creëren: een nieuw programma kunnen maken

2.4 Welke leerniveaus komen er in de methodes aan bod?

Voor iedere methode hebben we voor het onderdeel programmeren per hoofdstuk de oefenvragen en opdrachten ingedeeld in de zeven categorieën. Zo bepalen we op welke leerniveaus de methodes een beroep doen.

De resultaten van de niveau-indeling van de opdrachten staan weergegeven in figuur 3.



Figuur 3: Typering van opgaven in twee programmeermethodes (Enigma en Fundament) aan de hand van de leerniveaus uit figuur 2

Figuur 3 laat zien dat er in *Fundament* veel vragen staan uit de niveaus 1 en 3. Leerlingen moeten een tekst lezen en er vervolgens vragen over beantwoorden. De antwoorden kunnen ze uit de tekst halen. Vervolgens moeten ze de opgedane kennis toepassen in een herkenbare situatie. Vragen uit niveau 2, die inspelen op het begrijpen van de stof, zijn er niet veel. Hier zit dus een hiaat, want volgens de taxonomie van Bloom kun je kennis niet toepassen als je de leerstof alleen maar kent (kunt reproduceren) en niet begrijpt.

Enigma biedt vooral veel kennis- en begripsvragen (niveaus 1 en 2).

In beide methodes komen de opdrachten uit niveau 4, het toepassen van de opgedane kennis in een minder bekende situatie, veel minder voor. Van de hoogste niveaus van Bloom zijn in de lesmethodes geen opdrachten te vinden. De leerlingen worden niet aangespoord om te analyseren, te evalueren of zelf programma's te maken.

2.5 Waaraan opdrachten verder moeten voldoen

De vakliteratuur over leren programmeren bevat nog een aantal eisen waaraan lesmethodes moeten voldoen. We noemen er vier en passen die toe op *Fundament* en *Enigma*.

- *De leerling leert een computerprogramma lezen en beschrijven in eigen woorden*

Als hij dit kan, betekent dit dat hij begrijpt hoe het programma in elkaar zit. Kan hij dit niet, dan kan hij ook zelf geen programma schrijven (Lopez, 2008). *Fundament* geeft veel voorbeeldprogramma's, maar geen opdrachten om in eigen woorden te beschrijven wat een programma doet (de ontbrekende begripsoopdrachten uit niveau 2). *Enigma* geeft wel voorbeeldprogramma's met bijbehorende beschrijfoopdrachten.

- *De leerling krijgt voorbeelden, toepassingen en opgaven aangeboden*

Aan alleen tekst en uitleg heb je niets; om tot herkenning en begrip te komen heb je visuele voorbeelden nodig en moet je meteen toepassingen kunnen uitproberen (Merriënboer, 1990). *Fundament* geeft veel voorbeelden en kleine opdrachten in een context, maar geen grote opdrachten zoals het ontwikkelen van een programma in een ruime context. Ook *Enigma* biedt geen groot project in een context aan – en dat terwijl er zowel in het vwo- als het havo-programma 60 uur voor een project is gereserveerd.

- *De leerlingen werken samen aan opdrachten*

Als leerlingen programmastructuren, oplossingen en fouten met elkaar bespreken, gaan ze de leerstof beter begrijpen (McDowell, 2006). *Fundament* geeft alleen korte, eenvoudige opdrachten en biedt de leerlingen daarmee nauwelijks de mogelijkheid om vergaand samen te werken. *Enigma* raadt nergens expliciet aan om een opdracht samen met een andere leerling te doen, maar geeft wel grotere opdrachten die geschikt zijn voor teamwerk.

- *Opdrachten moedigen afhakers aan en dagen doordouwers uit*

Als een afhaker een opdracht niet begrijpt, doet hij helemaal niets meer. Hij denkt dat hij de opdracht niet aankan omdat hij niet slim genoeg is. Een doordouwer gaat juist wel verder en ontwikkelt een eigen oplossingsstrategie.

Twee dingen zijn van belang. Ten eerste dat een leraar of lesmethode het zelfvertrouwen van een afhaker kan vergroten door hem te laten inzien dat hij fouten mag (en zelfs moet) maken. En ten tweede dat de leerstof zo gedifferentieerd moet zijn dat de afhakers genoeg hulp krijgen en de doordouwers genoeg uitdaging (Bennedsen, 2008). In *Enigma* en *Fundament* staan geen opdrachten die het leergedrag van de leerling willen veranderen, geen opdrachten die ingaan op het maken en analyseren van fouten. En beide methodes bieden, met alleen maar opdrachten in de laagste leerniveaus, geen uitdaging voor de doordouwers, de slimme, snelle leerlingen.

3 Wat leraren en leerlingen van programmeren vinden

3.1 Vier scholen

Om uit te vinden hoe leraren en leerlingen in de praktijk met de lesmethodes werken en wat de leeropbrengsten van de methodes zijn, riepen we de hulp in van vier leraren op vier vwo-scholen. We interviewden hen en woonden hun lessen bij. Twee leraren maakten gebruik van Enigma en twee van Fundament.

We wilden ook weten wat de leerlingen op de vier scholen vonden van de lessen informatica en meer in het bijzonder van de lessen programmeren. Waarom hadden ze informatica gekozen? Wat wisten ze al van programmeren? Wat verwachtten ze van de lessen en kwamen de verwachtingen uit? Vonden ze de lessen leuk, interessant, makkelijk, moeilijk? Vielen de cijfers voor de toetsen mee of tegen?

3.2 De werkwijze van de leraren

Van de vier leraren die we geobserveerd en geïnterviewd hebben, heeft er maar één een afgeronde Computer Science Education opleiding. Twee anderen studeren nog aan de lerarenopleiding informatica.

De vier leraren blijken heel verschillend om te gaan met de lesmethodes: achtergrondmateriaal dat eventueel geraadpleegd kan worden, of leidend bij de lessen. Twee van hen geven de leerlingen alleen maar zelfgemaakte opdrachten. De leerlingen lezen de hoofdstukken uit de methode voor de theoretische achtergrond. De andere twee leraren volgen de methode wel, maar maken hun eigen keuzes uit de opdrachten in het boek. Soms vullen ze die aan met afsluitende opdrachten die ze zelf gemaakt hebben.

Op elke school, bij elke leraar, wordt iets anders aangeboden. Er is weinig samenhang tussen lessen, methodes en toetsen. De leerlingen volgen geen vast lesprogramma aan de hand van een methode. Er is geen eenheid in het programmeeronderwijs. Dit is wat je de Toren-van-Babelsituatie zou kunnen noemen.

Deze situatie is verklaarbaar vanuit de geschiedenis van het Nederlandse informaticaonderwijs. De eerste leraren waren geschoold in de CODI-opleiding – een bijscholingscursus, geen volwaardige opleiding. Daarna zijn de leraren zelf verder gegaan met lessen bedenken en lesmateriaal ontwikkelen, individueel of in auteursteams. Dit heeft geleid tot de Toren-van-Babelsituatie: er is

op verschillende scholen een enorme diversiteit aan lesmateriaal, opdrachten en lesdoelen. Ook de programmeertalen en de ontwikkelomgevingen waarmee leraren werken variëren. Natuurlijk getuigt dit van creativiteit, maar het is de vraag of je met deze wildgroei wel effectief, doelgericht en samenhangend programmeeronderwijs kunt ontwikkelen.

3.3 De mening van de leerlingen

Aan het begin en aan het einde van de lesperiode namen we bij de leerlingen een attitudetest af om te weten te komen wat ze van het onderwijs verwachtten en vonden. Voordat het programmeeronderwijs begon, vroegen we de leerlingen bijvoorbeeld waarom ze informatica hadden gekozen, of ze dachten dat het bruikbaar was voor hun vervolgstudie, wat ze er al van wisten en of ze thuis veel met computers bezig waren. We maten de houding van de leerlingen ten opzichte van het programmeeronderwijs met vragen als: wat verwacht je van dit vakonderdeel? Denk je dat het leuk is? Denk je dat het moeilijk is? Heb je er vertrouwen in dat je het aankan? Denk je dat jongens er beter in zijn dan meisjes?

De verwachtingen van de leerlingen voorafgaand aan de lessen waren positief. Ze verwachtten dat het leuk zou zijn en makkelijk, net als de andere onderdelen van informatica. Ze verwachtten dat het nuttig zou zijn voor hun vervolgstudie en dachten niet dat jongens er beter in zouden zijn dan meisjes. Wel waren er zeer grote verschillen tussen de instroomniveaus en de vaardigheden van leerlingen.

Gedurende de lesperiode interviewden we ook een aantal leerlingen. Er waren er weinig die informatica moeilijk vonden. Ze vonden het een leuk vak, ze waardeerden de leraren en de lessen. Ook over de lesmethode waren ze tevreden. Ze vonden programmeren relatief eenvoudig en hoefden er buiten de lessen om weinig voor te doen. We kregen zelfs de indruk dat deze leerlingen niet genoeg uitgedaagd werden.

Uit de attitudetesten aan het einde van de periode kregen we een heel ander, veel negatiever beeld. Dit kan samenhangen met de tegenvallende toetsresultaten (die besproken worden in hoofdstuk 4). 50% van de leerlingen concludeerde, nadat ze het programmeeronderwijs gevolgd hadden, dat het onderdeel moeilijker was en minder leuk dan ze hadden verwacht. Opvallend was dat de leerlingen die werkten met Enigma na de lessen dachten dat jongens beter konden programmeren dan meisjes. Vooraf dachten ze dat niet. Over het algemeen was het zelfvertrouwen van de leerlingen afgenomen.

Er lijkt een verband te zijn tussen de Toren-van-Babelsituatie, de tegenval-lende toetsresultaten en de veranderde houding van de leerlingen. De lessen staan los van de oorspronkelijke lesmethodes en sluiten hierdoor niet aan op de toetsen. De leerlingen ervaren de door de leraren bedachte lessen wel als leuk en makkelijk, maar halen slechte cijfers op de toetsen. Hun houding tegen-over het vak verandert en ze worden onzeker over hun vaardigheden.

4 De leeropbrengst van het programmeeronderwijs

4.1 De toetsen

Om inzicht te krijgen in de vorderingen van de leerlingen namen we twee toetsen af. De inhoud van de eerste – tussentijdse – toets stemden we af op de tot dan toe behandelde stof. De toetsen bestonden uit verschillende typen opgaven:

(1) Herkennen van programmeerconstructies

Opdracht: we lieten een programma (= omschrijving in programmeertaal) voor een klasse zien, de leerling moest deze herkennen (niveau 1). Een voorbeeld van zo'n opdracht staat hieronder beschreven.

Resultaat: op alle scholen gaat het herkennen van programmeerconstructies goed. Op de twee scholen die gebruikmaken van *Fundament* herkennen de leerlingen programmeerconstructies beter in de eindtoets dan in de tussentijdse toets. Dat ligt aan de opbouw van de lesstof.

(2) Herkennen en beschrijven van fouten

Opdracht: we lieten een programma voor een klasse zien met een aantal fouten, de leerling moest aangeven in welke regels fouten voorkwamen en beschrijven hoe de fouten verbeterd konden worden (niveau 1, gekoppeld aan niveau 2).

Resultaat: bij het herkennen van fouten treden er verschillen op, maar die zijn niet te herleiden tot de verschillende lesmethodes. Op de ene school waar ze *Fundament* gebruiken, herkent ongeveer 60% van de leerlingen de fouten, maar op de andere school waar ze deze methode gebruiken herkent maar 36% van de leerlingen de fouten. Ook onder *Enigma*-gebruikers zijn er verschillen.

(3) Omschrijven van begrippen

Opdracht: de leerling moest een aantal programmeerbegrippen (zoals 'programma', 'klasse' of 'attribuut') in eigen woorden omschrijven (niveau 2).

Resultaat: de prestaties bij het omschrijven van begrippen zijn voldoende en blijven gedurende de lesperiode op de *Fundament*-scholen ongeveer gelijk: de tussentijdse toets en de eindtoets laten dezelfde resultaten zien. Ook de leerlingen van de *Enigma*-scholen scoren goed op het omschrijven van begrippen.

(4) Zelf een code schrijven

Opdracht: de leerling moest zelf een programma in programmeertaal schrijven of veranderen (niveau 3 en 4).

Resultaat: zelf een programma schrijven vinden de leerlingen van alle scholen lastig. Minder dan de helft van de leerlingen van de *Fundament*-scholen beheerst dit goed. Op de *Enigma*-scholen kunnen we weinig lijn ontdekken in de resultaten voor het zelf schrijven. Duidelijk is wel dat op meerdere onderdelen het toepassen van de leerstof de leerlingen niet goed afgaat.

```
8 import java.util.*;
9
10 public class Bibliotheek
11 {
12     private String naam;
13     private String website;
14     private int aantalBoeken;
15
16     private Adres adres;
17     private ArrayList<Boek> boeken;
18
19     public Bibliotheek( String nm, String web, int nrBoeken, String straat, int nummer, String plaats)
20     {
21         naam = nm;
22         website = web;
23         aantalBoeken = nrBoeken;
24
25         adres = new Adres(straat, nummer, plaats);
26         boeken = new ArrayList<Boek>();
27     }
28
29     public String getAdres()
30     {
31         String straat = adres.getStraatnaam();
32         int nummer = adres.getHuisnummer();
33         String plaats = adres.getWoonplaats();
34
35         return "Straat: " + straat + " Huisnummer: " + nummer + " Woonplaats: " + plaats;
36     }
37
38     public String getNaam()
39     {
40         return naam;
41     }
42
43     public void printBoekenLijst()
44     {
45         System.out.println("Dit is een lijst met alle boeken uit deze bibliotheek");
46         for( Boek boek : boeken)
47         {
48             System.out.println("ISBN: " + boek.getIsbn() + " Titel: " + boek.getTitel() + " Auteur: " + boek.getAuteur());
49         }
50         System.out.println("Einde van de lijst");
51     }
52
53     public void voegBoekToe(int isbn, String titel, String auteurNaam)
54     {
55         Boek boek;
56         boek = new Boek(isbn, titel, auteurNaam);
57         boeken.add(boek);
58     }
59 }
```

Figuur 4: Voorbeeld van een programma voor een opgave uit de eindtoets. Leerlingen moeten programmeerconstructies herkennen en begrijpen, en de geleerde begrippen kunnen toepassen.

4.2 De relatie tussen de methodes en de toetsresultaten

Over de relatie tussen de lesmethodes en de leeropbrengsten kunnen we in elk geval zeggen dat er weinig verschil te ontdekken is tussen *Enigma* en *Fundament*. De resultaten op scholen die dezelfde methode gebruiken, lopen net zover uiteen als de resultaten op scholen die verschillende methodes gebruiken. De vergelijking tussen de methodes is ook eigenlijk ondoenlijk door de Toren-van-Babelsituatie: niet alle leraren houden zich aan de lesmethodes en iedereen zet eigen lesmateriaal en opdrachten in. Wat er in de methodes staat, is lang niet altijd wat er in de lessen aan de orde komt.

Wat de toetsresultaten in ieder geval wel laten zien is dat leeropbrengsten tegenvallen: veel leerlingen vinden het herkennen en beschrijven van fouten moeilijk en de meeste leerlingen hebben moeite met het toepassen van de lesstof. We zagen in hoofdstuk 2 al dat de beide methodes weinig aandacht schenken aan toepassingsopdrachten. Aan de hogere Bloomniveaus (analyseren, evalueren en creëren) wordt helemaal geen aandacht besteed, terwijl daaruit juist de leukere opdrachten zouden kunnen komen die leerlingen motiveren.

5 Conclusie

In dit verkennend onderzoek hebben we het informaticaonderwijs, en in het bijzonder het programmeeronderwijs, geëvalueerd. Het blijkt dat het programmeeronderwijs nog niet de opbrengsten oplevert die beoogd worden. Leerlingen kunnen op meerdere onderdelen de leerstof niet goed toepassen. Ze hebben geleerd *over* programmeren, maar ze hebben niet geleerd *om* te programmeren. De leerlingen vinden na afloop van de lessen het programmeren ook minder leuk en moeilijker dan ze vooraf verwachtten. Dit heeft meerdere oorzaken:

1. Globale, vrijblijvende leerdoelen bieden weinig houvast

De leerdoelen voor het programmeeronderwijs zijn globaal en vrijblijvend, daardoor is er een grote variatie in het informaticaonderwijs ontstaan. Heldere, expliciete eindtermen kunnen zorgen voor meer eenheid in het lesmateriaal en bieden meer houvast voor makers van lesmethodes en standaardtoetsen.

2. Methodes beperken zich tot lagere leerniveaus

De methodes *Fundament* en *Enigma* beperken zich vooral tot de lagere leerniveaus: begrijpen, onthouden, direct toepassen. Opdrachten waarin leerlingen moeten analyseren, evalueren en creëren komen niet of nauwelijks in beide methodes voor, terwijl daaruit juist de leukere opdrachten zouden kunnen komen die leerlingen – en dan vooral de doordouwers (de snelle, slimme leerlingen) – motiveren en uitdagen. Hierdoor komen leerlingen nauwelijks toe aan het creatieve, constructieve aspect van het vak: ontwerpen en bouwen, terwijl dat jonge mensen juist aanspreekt. Voor de afhakers (de leerlingen met weinig zelfvertrouwen) bieden de methodes geen houvast.

3. De lessen hebben weinig samenhang met de methode

Op elke school, bij elke leraar, wordt iets anders aangeboden. Het is moeilijk om informaticaleraren te vinden die een lesmethode intensief gebruiken. De lessen hangen weinig samen met de methode en met de toetsen. De leerlingen volgen geen vast lesprogramma aan de hand van een methode. Dit hebben we de Toren-van-Babelsituatie genoemd. Meer eenheid en samenhang in het lesmateriaal leidt tot beter uitgewerkte lessen met betere resultaten. Vaak maken leraren hun eigen lesmateriaal in verschillende programmeertalen en ontwikkelomgevingen, maar veel leraren zijn niet voldoende opgeleid om goed materiaal te kunnen ontwikkelen.

4. Leerlingen hebben verkeerde verwachtingen

Met betere voorlichting weten leerlingen beter waar ze aan beginnen. Ze moeten meteen begrijpen dat ze alleen met doorzettingsvermogen (vallen en opstaan: fouten maken en analyseren) en tijdsinvestering kunnen leren programmeren.

6 Meer weten?

6.1 Literatuur

- Barendsen, E., Franquinet, R., Leijtens, R. & Reinders, H. (2007). *Enigma demo Informaticacommunity: voor de tweede fase verwerkingsboek*. Beschikbaar op: <http://enigma.cs.ru.nl/wp-content/uploads/2009/03/demo-verwerkingsboek-v.pdf>.
- Bennedsen, J. & Caspersen, M.E. (2008). Exposing the programming process. In: *Reflections on the Teaching of Programming*, Lecture Notes in Computer Science, 4821: 6–16.
- Bloom B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. New York: David McKay Co Inc.
- Koldenhof, E., Jeuring J. & Ruth, S. (2011), *Rendement van objectgericht programmeeronderwijs*. Utrecht: Universiteit Utrecht.
- Laan, J. van der (red.) (2007). *Fundament Informatica: module VI tot en met VIII: ICT voor de tweede fase*. Bodegraven: Instruct.
- Lopez, M., Whalley, J., Robbins, P. & Raymond, L. (2008). Relationships between reading, tracing and writing skills in introductory programming. In: *Proceeding of the Fourth international Workshop on Computing Education Research*, ICER '08 (101–112). ACM.
- Merriënboer, J.G. van & Paas, F.G.W.C. (1990). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behaviour*, 6: 273–289.
- Nationaal expertisecentrum leerplanontwikkeling (SLO) (2009). *Eindexamenprogramma informatica HAVO/VWO. 2009*. Enschede: SLO.
- Perkins, D.N., Hancock, C., Hobbs, R., Martin, G. & Simmons, R. (1986). Conditions of learning in novice programmers. In: E. Soloway & J.C. Spohrer (Red.), *Studying the novice programmer* (261–279). Hillsdale, NJ: Lawrence Erlbaum.
- Proulx, V.K. (2000). Programming patterns and design patterns in the introductory computer science course. In: *Proceedings of the 31st SIGCSE technical symposium on Computer science education*, SIGCSE '00 (80–84). ACM.
- Schmidt, V. (2007). *Handreiking schoolexamen informatica HAVO/VWO*. Enschede: SLO.
- Zwaneveld, B., Moor, R. de, Perrenet, J. & Diepen, N. van (2009). Positie van het vak informatica in havo/vwo. *Tijdschrift voor didactiek der β-wetenschappen*, 26 (1-2): 37-54.

6.2 Over het onderzoek

Deze publicatie is gebaseerd op het onderzoek *Rendement van objectgeoriënteerd programmeren*. U kunt het onderzoeksverslag raadplegen op: <http://onderzoek.kennisnet.nl/onderzoeken-totaal/programmeeronderwijs>.

6.3 Over de auteurs

Ernst Koldenhof heeft de studie natuurkunde in Utrecht in 1985 afgerond, en daarna tot 1991 promotieonderzoek verricht op het Kernfysisch Versneller Instituut in Groningen. Daarna heeft hij tot 2009 verschillende ICT functies uitgeoefend bij Lorjé, Simac en Philips Halfgeleiders, later NXP. Vanaf 2009 is hij als docent informatica werkzaam op O.R.S. Lek en Linge in Culemborg en sinds 2009 volgt hij de masteropleiding Science Teacher Education op de universiteit van Utrecht.

Johan Jeuring is hoogleraar Softwaretechnologie aan de faculteit Informatica van de Open Universiteit, en hoofddocent en teaching fellow bij de Universiteit Utrecht. In zijn onderzoek kijkt hij naar mogelijkheden om programmeertalen te verbeteren, zodat ze het schrijven van goede programma's beter ondersteunen, en naar technologieën voor het ontwikkelen van elektronische leeromgevingen, met name voor wiskunde en informatica. Daarbij kijkt hij vooral naar het geven van hints en feedback aan studenten. De door Jeuring's groep ontwikkelde feedback-services zijn de afgelopen jaren door duizenden leerlingen gebruikt. Zijn onderwijstaak bestaat tegenwoordig vooral uit het begeleiden van onderwijskundige vernieuwingen binnen het hoger onderwijs.

Sandra Ruth studeert informatica aan de Universiteit Utrecht en is in de eindfase van haar Bachelor. Zij deed binnen haar opleiding ervaring op met de praktische kant van het onderwijs door het lopen van een stage bij de sectie Informatica van een HAVO/VWO school in het voortgezet onderwijs.

6.4 Een vraag stellen

De afdeling Onderzoek van Kennisnet kan specifieke vragen over dit onderzoek beantwoorden. Mail naar onderzoek@kennisnet.nl of bel naar 0800-321 22 33.

Colofon

Zelf programmeren. Een toets voor het informaticaonderwijs

© Kennisnet, Zoetermeer
December 2011

Opdrachtgever:
Stichting Kennisnet, Zoetermeer

ISBN: 978-90-77647-53-0

Auteurs:
Ernst Koldenhof, Johan Jeuring en Sandra Ruth (Universiteit Utrecht)

Redactie
Het Laatste Woord, Bennekom

Vormgeving
Tappan Communicatie, Den Haag

Druk
OBT, Den Haag



Naamsvermelding-NietCommercieel-GeenAfgeleideWerken 2.5 Nederland

De gebruiker mag:

- het werk kopiëren, verspreiden, tonen en op en uitvoeren onder de volgende voorwaarden:
 -  Naamsvermelding. De gebruiker dient bij het werk de naam van Kennisnet te vermelden.
 -  Niet-commercieel. De gebruiker mag het werk niet voor commerciële doeleinden gebruiken.
 -  Geen Afgeleide werken. De gebruiker mag het werk niet bewerken.
- Bij hergebruik of verspreiding dient de gebruiker de licentievoorwaarden van dit werk kenbaar te maken aan derden.
- De gebruiker mag uitsluitend afstand doen van een of meerdere van deze voorwaarden met voorafgaande toestemming van Kennisnet.

Het voorgaande laat de wettelijke beperkingen op de intellectuele eigendomsrechten onverlet.
(www.creativecommons.org/licenses)

Dit is een publicatie van Stichting Kennisnet. kennisnet.nl

Kennisnet Onderzoeksreeks

Wat weten we uit wetenschappelijk onderzoek over ict in het onderwijs en hoe kunnen scholen samen met onderzoekers voortbouwen op beschikbare resultaten uit eerder uitgevoerd onderzoek?

De Kennisnet Onderzoeksreeks *Ict in het onderwijs* heeft als doel een verzamelplaats te zijn voor antwoorden op deze vragen. Daarvoor wordt gebruik gemaakt van de praktijkervaringen van onderwijsprofessionals en resultaten uit wetenschappelijk onderzoek. Deze reeks is bedoeld voor management en leraren in het onderwijs en voor instellingen en organisaties die het onderwijs ondersteunen bij effectief en efficiënt gebruik van ict.

2008

- #1 Kennis van Waarde Maken
- #2 Leren met meer effect
- #3 Ict werkt in het vmbo
- #4 Games in het (v)mbo
- #5 Web 2 in de BVE
- #6 Digitale schoolborden in het PO
- #7 Speciaal onderwijs levert maatwerk met ict
- #8 Opbrengsten van ict-projecten
- #9 Leren in Second Life
- #10 HomoZappiens@Schonenvaart.mbo

2010

- #21 Zelfstandig leren rekenen met het digibord
- #22 Leren van moderne vreemde talen
- #23 Opbrengsten van Leren met meer effect
- #24 Meerwaarde van het digitale schoolbord
- #25 Effecten van games
- #26 Maak kennis met TPACK
- #27 Duurzame onderwijsvernieuwing
- #28 De prijs van digitaal leermateriaal
- #29 Een digitaal klassenboek
- #30 Leren met je mobiel

2009

- #11 Web 2.0 als leermiddel
- #12 De betrouwbaarheid van internetbronnen
- #13 Leren met meer effect: de onderzoeksresultaten
- #14 Samen Engels Leren Spreken
- #15 Taalontwikkeling van jonge kinderen
- #16 Digitaal leermateriaal taalonderwijs PO
- #17 Jongeren en interactieve media
- #18 Essays over bruikbaar digitaal leermateriaal
- #19 Computersimulaties in het VO
- #20 Eerst onderwijsvisie, dan techniek

2011

- #31 Opbrengsten van EXPO
- #32 Zes voordelen van ict voor het mbo
- #33 Webquests
- #34 Ict en rekenen in het basisonderwijs
- #35 Synchron coachen
- #36 **Programmeeronderwijs**

Stichting Kennisnet

Postadres	Bezoekadres	T (0800) 321 22 33
Postbus 778	Paletsingel 32	E info@kennisnet.nl
2700 AT Zoetermeer	2718 NT Zoetermeer	W kennisnet.nl