Computing-onderwijs in de praktijk

Wat kunnen we leren van de Britten?

Kennisnet

Voorwoord

'het Britse computing-curriculum werd in september 2014 verplicht gesteld voor alle openbare scholen.'

Kennisnet krijgt de laatste tijd steeds vaker vragen van scholen in het funderend onderwijs over programmeren. Wat kun je als school doen? We werden mede-afzender van de Codeweek, we maakten Codekinderen.nl met lessuggesties, en we initieerden een speciale aflevering van Donald Duck over programmeren. Allemaal bedoeld om leerkrachten en leerlingen te inspireren.

Inmiddels hebben sommige scholen aan inspiratie niet meer genoeg. Ze willen 'programmeren' in hun onderwijs integreren. Andere scholen vinden 'programmeren' te beperkt, en willen hun leerlingen 'digitaal geletterd' maken. Maar hoe doe je dat?

In Groot-Brittannië is programmeren onderdeel van het nieuwe (en verplichte) computing-onderwijs aldaar. We hebben dit onderwijs onder de loep genomen en vertaald naar lessen waar Nederlandse scholen verder mee kunnen.

Tot slot: het Britse computing-curriculum (onderdeel van het algemene National Curriculum) werd in september 2014 verplicht gesteld voor alle openbare scholen. In Nederland werkt het natuurlijk anders: ons onderwijssysteem kent geen verplichte curricula. Maar áls je als basisschool of middelbare school iets wilt met 'computational thinking', kun je je voordeel doen met de ervaringen uit Groot-Brittannië. Daarover gaat dit rapport.

Toine Maes, directeur Kennisnet

1. Inleiding

"Onze samenleving en economie veranderen door de opmars van digitalisering en nieuwe technologie razendsnel. Herzieningen van onderwijsprogramma's zijn op alle niveaus broodnodig." Aldus de economen Willem Vermeend en Rick van der Ploeg, in hun artikel *Onderwijs moet op de schop*, naar aanleiding van de lancering van het Platform Onderwijs 2032 (*De Telegraaf*, 16 februari 2015).[1]

Maar ook vanuit niet-economische hoek zijn er zulke signalen. De KNAW (Koninklijke Nederlandse Academie van Wetenschappen) sprak al in 2012 haar bezorgdheid uit, in het rapport *Digitale geletterdheid in het voortgezet onderwijs:* "Het [informaticavak] negeert de enorme ontwikkelingen in de digitalisering van informatie en communicatie en de impact daarvan in de afgelopen 20 jaar. [...] Het vak is inhoudelijk uit de tijd."[2] Of, de SLO (Nationaal Expertisecentrum Leerplanontwikkeling), in het rapport *Informatica in de bovenbouw havo/vwo (2014):* "Er moet een nieuw examenprogramma komen dat beter aansluit op vervolgopleidingen en leraren moeten bijgeschoold worden."[3]

'Het huidige ict-onderwijs, zo vinden de critici, levert te weinig inzicht, kunde en vaardigheden.'

Het huidige ict-onderwijs, zo vinden de critici, levert te weinig inzicht, kunde en vaardigheden. Bovendien, zo blijkt uit een recent onderzoek onder 14-jarigen, dragen de huidige lessen over computers en digitale media niet of nauwelijks bij aan hun digitale geletterdheid. Volgens dit onderzoek (uitgevoerd door de Universiteit Twente, in opdracht van Kennisnet), leren leerlingen op dit moment vooral *thuis* hoe ze effectief kunnen omgaan met computers en digitale informatie.[4]

Belangrijk om te constateren, bij al deze kritiek op het huidige ict-onderwijs, en de (impliciete of soms zelfs expliciete) boodschap om meer aandacht te besteden aan programmeren, is dat het daarbij niet alleen gaat om het opleiden van toekomstige programmeurs, maar vooral ook om ervoor te

zorgen dat kinderen en jongeren grip krijgen op de totaal gedigitaliseerde wereld om hen heen. Zoals het (verplichte) vak wiskunde ooit bedoeld was om te begrijpen hoe 'wiskunde' doorwerkt in bijna alle disciplines en wetenschappen, zo zou je nu 'computational thinking' moeten onderwijzen om te begrijpen hoe dagelijkse dingen als zoekmachines, routeplanners en wifi werken, is het idee.

Computing en computational thinking

Als antwoord op de behoefte aan verandering en verbetering, werd in september 2014 in Groot-Brittannië een grootscheeps nieuw curriculum uitgerold, onder de naam 'computing'. (Dat wil zeggen: het werd al drie jaar eerder geïntroduceerd, maar in 2014 officieel verplicht gesteld.)

Dit type onderwijs traint leerlingen in een manier van denken waarbij je een probleem en de mogelijke oplossingen leert formuleren in computer-termen, kortweg: *computational thinking*[5]. Je zou het 'begrijpend programmeren' kunnen noemen, vergelijkbaar met 'begrijpend lezen'.

Britse kinderen leren dit nu vanaf hun 5e jaar. Als ze op hun 11e het primair onderwijs verlaten, kennen ze twee computertalen, kunnen ze robots, smartphones en veiligheidssystemen besturen, en beheersen ze de basisprincipes van netwerken en computational thinking. In het voortgezet onderwijs bouwen ze hun kennis verder uit, begeleid door natuurkundedocenten, wiskundedocenten, en IT-specialisten.

Moeten alle kinderen dan programmeur worden, zoals sommige critici vrezen? "Nee, dat is niet de bedoeling," zegt Bill Mitchell, onderwijsdirecteur bij het British Computing Centre (BCS). "Maar wel om het creatieve, oplossingsgerichte vermogen van kinderen aan te scherpen, en hun kennis van digitale systemen te vergroten."[6]

'Desondanks zijn er ook in Nederland specifieke scholen, met bevlogen docenten'

Koplopers in Nederland

In veel Europese landen leren scholieren al jong programmeren, zodat ze al doende de grondbeginselen van informatica en mediawijsheid tot zich nemen, zo blijkt uit een recente publicatie van *European Schoolnet*.[7] Nederland is volgens dit rapport de hekkensluiter, samen met België en Frankrijk.

Maar desondanks zijn er ook in Nederland specifieke scholen, met bevlogen docenten, waar al computing-lessen worden gegeven. Steeds meer zelfs, zo blijkt uit de vele vragen van scholen aan Kennisnet over programmeren. Het verst gevorderd zijn de scholen die aangesloten zijn bij bestuurlijke verbanden als Oponoa (6 scholen), Fier (17 scholen) en Ooada (22 scholen). De desbetreffende schoolbesturen stelden speciale docenten aan, en dienden een versnellingsvraag in bij de PO-Raad, om uit te zoeken hoe dit type onderwijs versterkt kan worden.[8]

Tegengeluiden

Anderzijds zijn er ook tegengeluiden, van onderwijsdeskundigen die computing-onderwijs geen goed idee vinden, zeker als het verplicht zou worden. Onderwijs-columniste Aleid Truijens verwoordde het als volgt:

"Latijn en Grieks zijn niet verplicht. Niemand hoeft naar een gymnasium en je kunt het VWO halen zonder die vakken. Straks moeten kinderen, ook op het gymnasium, wel allemaal verplicht leren programmeren? Waarom? Omdat we allemaal dagelijks computers en tablets gebruiken? Rare redenering. We zitten ook allemaal op stoelen, daarom hoeven we toch geen stoel te kunnen timmeren? We rijden in auto's, maar niet iedereen leert hoe je zo'n dingt bouwt of hoe de onderdelen werken. Goddank hoeft dat niet. Ik maak dankbaar gebruik van de kunde van programmeurs, tandartsen, monteurs, muzikanten, timmerlieden en tuinmannen."[9]

Merk op dat 'computing' hier dus versmald wordt tot 'programmeren'. En dat we in Nederland geen verplichte curricula hebben (zoals in Groot Brittannië).

Centrale vraag

Maar áls een school aan de slag zou willen met computing, hoe moet dat dan? Wie moet het uitvoeren? Waarom, met welke middelen en volgens welk leerplan?

Op die vragen geeft dit rapport antwoorden, gebaseerd op de ervaringen in Groot-Brittannië, waar men al flink gevorderd is. Vandaar de centrale vraag: wat kunnen we leren van de Britten?

De keus voor Groot-Brittannië

In Groot-Brittannië werd in 2014 een compleet nieuw curriculum voor het informaticaonderwijs geïntroduceerd. De verwachtingen zijn hooggespannen. "Door van 5-jarigen te eisen dat ze zich met computerwetenschap bezighouden, is Engeland in één klap een land als de Verenigde Staten gepasseerd," aldus The Economist, eind september 2014. "De EU denkt dat Engeland een model voor zijn buren zal worden. Aziatische landen houden ons goed in de gaten."[10]

'De EU denkt dat Engeland een model voor zijn buren zal worden.'

Ook andere Europese landen spannen zich in; met name Estland en Griekenland gaan zeer grondig te werk. Toch zijn er goede redenen om het Britse computing-curriculum als uitgangspunt te nemen. Zo liggen de Britse onderwijssituatie, het docentschap en de lescultuur dichter bij de onze dan die van Estland en Griekenland. Bovendien is er veel meer en beter materiaal beschikbaar, zoals rapporten, interviews, voorbeeldlessen, praktijkvoorbeelden en verhalen van docenten. Dat materiaal is voor een groot deel verzameld door de grassroots-organisatie Computing at School (waarover later meer), die het nieuwe curriculum heeft begeleid en – niet onbelangrijk – zeer benaderbaar en behulpzaam is.

In dit rapport zullen we ons dus voornamelijk richten op datgene wat we van Groot-Brittannië kunnen leren. Waar nodig zullen we interessante ervaringen, ideeën en praktijken uit andere Europese landen echter niet onvermeld laten.

Werkwijze

Dit rapport is gebaseerd op literatuuronderzoek (rapporten, nota's, artikelen), en eigen interviews met Britse computing-docenten. We benaderden deze docenten via de organisatie *Computing at School* (CAS); zie onder.

Voor het samenstellen van de definitieve rapporttekst is dankbaar gebruik gemaakt van de commentaren van SLO (Nationaal Expertisecentrum Leerplan Ontwikkeling), Nederlandse onderwijsbestuurders, docenten, en andere betrokkenen uit het onderwijsveld.

'Computing at School' (CAS)

Een belangrijke bron was – zoals gezegd – *Computing at School* (CAS), een Britse organisatie die veel voor elkaar heeft gekregen. Opgericht in 2008 als een grassroots-organisatie (oftewel: een organisatie die zijn oorsprong heeft op de werkvloer, dus niet van overheidswege is ingesteld), wist CAS een succesvolle lobby op te zetten om computing een veel prominentere plaats te geven in het Britse schoolsysteem. Daarnaast wist zij met behulp van bevriende organisaties een compleet curriculum samen te stellen en uit te rollen in het primaire en secundaire onderwijs. Ten slotte wist CAS een netwerk van docenten, wetenschappers en IT-medewerkers te creëren, waardoor docenten zich konden laten bijscholen, en zo de nieuwste inzichten en praktijken op het gebied van informatica konden opdoen.

Initiatiefnemer van CAS is Simon Peyton Jones (hoogleraar informatica), die nauwe banden onderhoudt met Microsoft. Onder zijn leiding wist CAS in enkele jaren uit te groeien tot een organisatie van meer dan 2000 leden,

waarvan het merendeel bestaat uit schooldocenten en informaticadocenten, aangevuld met IT-specialisten, software-deskundigen, onderzoekers, en pleitbezorgers uit diverse hoeken.

Vooral door toedoen van CAS – met steun van Microsoft en de Royal Society (Academy of Sciences) – besloot de Britse staatssecretaris van onderwijs, Michael Gove, het oude ict-curriculum te laten herschrijven en landelijk in te voeren.[11]

Overwegingen achter het nieuwe curriculum

Waarom moest het nieuwe – verplichte – vak computing in de plaats komen van het oude vak *Information and Communication Technologies?* Daarvoor waren drie belangrijke argumenten (geformuleerd door CAS-prominenten en nauw betrokkenen): achtereenvolgens ideologisch, onderwijskundig, en – uiteraard – economisch van aard.

Ideologisch – computing zou kinderen een betere grip op de toekomst geven. Ze komen terecht in een wereld van digitale systemen en genetwerkte computers, en moeten die niet alleen leren *gebruiken* maar ook leren *begrijpen* en *beheersen*. Dat wordt van ze verlangd in het hoger onderwijs, maar ook in de rest van hun carrière.
Onderwijskundig – computing is volgens CAS ook belangrijk voor andere schoolvakken. *Computational thinking* leert je hoe je problemen op een systematische wijze kunt oplossen, wat heuristiek is, hoe algoritmes werken, wat intelligentie is, hoe je gegevens kunt manipuleren, etc. “Bij natuurkunde leer je te denken over natuurkunde. Maar bij computing leer je te denken over het denken zelf,” volgens Bill Mitchell, onderwijsdirecteur bij het British Computing Centre (BCS). “Je moet bedenken hoe je een computer iets voor jou kan laten oplossen. Je leert heel wat vaardigheden die je in andere vakken ook kunt gebruiken.”[12]

Economisch – de kloof tussen het aantal IT-vacatures en het aantal afgestuurde vakmensen wordt steeds groter, ook al omdat de banen in de games/media/app-wereld steeds veeleisender worden. Er worden tekorten van duizenden (in Nederland: 7.000 tot 12.000) professionals verwacht, ook als de economie nauwelijks groeit.

Onderbouwing ontbreekt

Hoe steekhoudend de bovenstaande argumenten zijn, is nog een open vraag. “Er is geen overtuigend bewijs dat leren programmeren kinderen helpt om meer algemene probleemoplossende vaardigheden te ontwikkelen,” zegt David Buckingham, oprichter van het Britse *Centre for the Study of Children, Youth and Media*, en onderwijsprofessor aan het *Institute of Education* van de London University.” Ook het idee dat die verplichte lessen informatica-banen zal opleveren, is dubieus,” aldus Buckingham.[13]

‘Er is geen overtuigend bewijs dat leren programmeren kinderen helpt om meer algemene probleem-oplossende vaardigheden te ontwikkelen’

Er is inderdaad nog weinig onderzoek dat klip en klaar aantoont dat computing-onderwijs leervermogen, kennisverwerving en creatief denken stimuleert. Alleen de onderzoeken van Sze Yee Lye en Joyce Hwee Ling Koh op basisscholen in Singapore geven een indicatie dat het werkt.[14]

De centrale rol van programmeren

In het Britse computing-curriculum staat programmeren centraal. Leer je programmeren, zo vinden de Britten, dan leer je tegelijkertijd iets over al die andere aspecten van de digitale netwerksamenleving: zoals hardware en software, besturingssystemen en protocollen, netwerken en apps, digitale risico’s en veiligheid. Programmeren is de locomotief die alle andere aspecten van de digitale wereld met zich meetrekt. Door kinderen te leren programmeren, leren ze ook alle andere aspecten van het moderne computergebruik kennen én beheersen, is de gedachte.

'Programmeren is dus geen doel, maar een middel.'

Programmeren is dus geen doel, maar een middel. "Verder is programmeren een uitstekende motivator," vindt Peyton Jones (CAS). "Niets moedigt leerlingen zo aan als het vooruitzicht om een computer naar je pijpen te kunnen laten dansen."

Terzijde: in plaats van 'programmeren' wordt soms ook de term 'coderen' gebruikt. Maar programmeren omvat meer dan alleen coderen. Programmeren is: een plan van aanpak maken, een algoritme bedenken, het algoritme vertalen naar opdrachten in een bepaalde programmeertaal (coderen), fouten opsporen en herstellen (debuggen), het programma documenteren (commentaar toevoegen) zodat andere programmeurs het kunnen begrijpen, etc. Coderen is dus een onderdeel van programmeren. Zoals programmeren weer een onderdeel is van software-ontwikkeling.

Terminologische verantwoording

Bij het samenstellen van dit rapport deed zich de vraag voor hoe je dit nieuwe type onderwijs zou moeten noemen. In de Britse onderwijswereld heeft men het bijvoorbeeld al lang niet meer over ict-onderwijs (de term is zelfs in de ban gedaan), maar over *computing*.

Omdat dit rapport voornamelijk over Groot-Brittannië gaat, zullen we de term **computing** overnemen. Met daarvan afgeleide samenstellingen als 'computing-onderwijs', 'computing-lessen', 'computing-docenten', en 'computing-coördinatoren'.

Aan 'computing' ligt het concept *computational thinking* ten grondslag. Als term voor een type onderwijs is dit echter minder geschikt (omdat het om een onderliggend concept gaat), en voor dagelijks gebruik is het zelfs ronduit onhandig. Woorden als 'computationele denk-lessen' of 'computationeel-denken-docenten' zullen het Nederlands niet snel veroveren.

Verder worden in dit rapport consequent de termen **primair onderwijs** en **secundair onderwijs** gebruikt om zo dicht mogelijk aan te sluiten bij de Britse onderwijspraktijk. (Denk: primary school en secondary school). Daar waar de Nederlandse situatie beschreven wordt (zoals de lessen die wij kunnen leren van de Britten), zullen vanzelfsprekend wél termen als 'basisonderwijs' en 'voortgezet onderwijs' gebruikt worden.

Zie ook: Bijlage 4.1 – *Termen en begrippen*.

2. De inrichting van het Britse computing-onderwijs

Voordat we de vraag kunnen beantwoorden welke lessen we kunnen trekken uit het Britse computing-onderwijs, is het belangrijk om te weten hoe het is ingericht. Hoe past het in hun *National Curriculum*? Hoe zien de lessen eruit, hoe zijn ze gestructureerd, wat is het didactisch model, hoe ziet de lespraktijk eruit, en hoe is de overgang verlopen?

2.1 Computing als onderdeel van het National Curriculum

Het Britse computing-onderwijs, dat in september 2014 van start ging, maakt deel uit van het *National Curriculum*. Dit nationale curriculum geldt voor alle scholen die door de staat gefinancierd worden. (Privé-scholen mogen hun eigen curriculum samenstellen, maar moeten zich wel aan algemene richtlijnen houden).

Het National Curriculum kent drie kernvakken (*core subjects*) – Engels, wiskunde en natuur-wetenschappen – en negen secundaire vakken (*foundation subjects*), waaronder 'computing'. Vóór september 2014 heette dit laatste vak *Information and Communication Technologies*. De key stages (sleutelfasen) 1 en 2 behelzen het primair onderwijs; de *key* stages 3 en 4 het secundair onderwijs. Zie Afbeelding 1. De vinkjes (✔) betekenen: 'verplicht'.

	Key stage 1	Key stage 2	Key stage 3	Key stage 4
Age	5 – 7	7 – 11	11 – 14	14 – 16
Year groups	1 – 2	3 – 6	7 – 9	10 – 11
Core subjects				
English	✓	✓	✓	✓
Mathematics	✓	✓	✓	✓
Science	✓	✓	✓	✓
Foundation subjects				
Art and design	✓	✓	✓	
Citizenship			✓	✓
Computing	✓	✓	✓	✓
Design and technology	✓	✓	✓	
Languages		✓	✓	
Geography	✓	✓	✓	
History	✓	✓	✓	
Music	✓	✓	✓	
Physical education	✓	✓	✓	✓

Afbeelding 1: Structure of the national curriculum

De crux van 'computing'

Crux van 'computing' is dat leerlingen het vermogen ontwikkelen om computerkennis en creativiteit te gebruiken om de wereld te begrijpen, en waar mogelijk (digitaal) naar hun hand te zetten. Kern daarvan zijn de grondbeginselen van het academische vak informatica, waardoor leerlingen thuisraken in *computational thinking*. Dat wil zeggen: begrijpen hoe digitale systemen werken en hoe je ze kunt inzetten door middel van programmeren.

Doelstellingen

Het Britse ministerie van onderwijs formuleert de doelstellingen van het nieuwe curriculum als volgt:

- "Het nationale curriculum voor computing-onderwijs is erop gericht om ervoor te zorgen dat alle leerlingen:
- de fundamentele principes en achterliggende ideeën van informatica kunnen begrijpen en toepassen, met inbegrip van abstractie, logica, algoritmes en data-representatie;
- problemen in computer-termen kunnen analyseren, en praktische ervaring opdoen in het schrijven van programma's waarmee dergelijke problemen opgelost kunnen worden;
- informatietechnologie kunnen begrijpen, evalueren en toepassen (met inbegrip van nieuwe of onbekende technologieën), om problemen analytisch op te kunnen lossen;
- verantwoordelijke, competente, zelfverzekerde en creatieve gebruikers worden van informatie- en communicatietechnologie."

2.2 Computing in het primair onderwijs

Het primair onderwijs in het Verenigd Koninkrijk omvat twee *key stages* (sleutelfases):

- sleutelfase 1 (*infant school*) voor 5- tot 7-jarigen;
- sleutelfase 2 (*junior school*) voor 7- tot 11-jarigen.

Hieronder volgen eerst de ministeriële richtlijnen voor beide fases. Daarna hoe het werkt in de praktijk.

Sleutelfase 1 (5-7 jaar)

- begrijpen wat algoritmes zijn, en hoe die omgezet kunnen worden naar programma's;
- begrijpen dat programma's met precieze en eenduidige instructies werken;
- eenvoudige programma's schrijven (coderen) en fouten daaruit verwijderen (debuggen);
- logische redeneringen gebruiken om het gedrag van eenvoudige programma's te voorspellen;
- digitale content creëren, organiseren, opslaan, manipuleren en weer binnenhalen;
- informatietechnologische toepassingen van buiten de school herkennen;
- de technologie veilig en respectvol gebruiken, en persoonlijke informatie beschermen;
- weten waar je naartoe moet als je hulp of ondersteuning nodig hebt, of wanneer je je zorgen maakt.

Sleutelfase 2 (7-11 jaar)

- programmeren (coderen en debuggen);
- natuurkundige systemen simuleren en besturen (met behulp van software);
- problemen oplossen door ze te ontleden in deelproblemen;
- sequentie, selectie en herhaling gebruiken in programmatuur, en werken met variabelen en verschillende vormen van invoer en uitvoer;
- logische redeneringen gebruiken, om uit te leggen hoe eenvoudige algoritmes werken;

- computernetwerken begrijpen (inclusief internet en world wide web);
- functies van netwerken begrijpen (communicatie en samenwerking);
- zoek-technologieën effectief gebruiken (begrijpen hoe de resultaten gerangschikt worden, en de resultaten kritisch beoordelen);
- bestaande software (inclusief internetdiensten) selecteren, gebruiken en combineren, op verschillende apparaten, om vooraf gegeven doelen te bereiken;
- gegevens verzamelen, analyseren, evalueren en presenteren;
- het veilig, respectvol en verantwoord gebruiken van technologie;
- aanvaardbaar en onaanvaardbaar (online-) gedrag leren onderscheiden;
- weten wat je moet doen als je bezorgd bent over dubieuze content of contacten.

Kern van het curriculum

Het nieuwe computing-curriculum komt er in essentie op neer dat leerlingen moeten thuisraken in de beginselen van informatietechnologie en *computational thinking*. Het Britse ministerie van onderwijs formuleert het zo:

"Het curriculum is zo opgebouwd dat in de eerste jaren op een speelse manier, meestal zonder schermen en digitale apparaten, de fundamenten van computational thinking worden bijgebracht. Bijvoorbeeld: wat zijn algoritmes, hoe werken ze, en wat kun je ermee in het dagelijks leven?"

In een online video[15] laat CAS-voorzitter Simon Peyton Jones zien hoe je jonge kinderen een sorteer-alogoritme kunt leren, zonder gebruik te maken van een computer. (Zie Afbeelding 2).

Afbeelding 2: Voorbeeld van een sorteeralgoritme zonder gebruikmaking van een computer. Zie: tedxexeter.com/2014/05/06/simon-peyton-jones-teaching-creative-computer-science

Hoewel de lesvormen speels en interactief zijn, gaat het dus al vanaf de eerste lessen over kernbegrippen als algoritmes, instructies, procedures, logische redeneringen, variabelen en sequenties. Met behulp van diverse werkvormen worden deze begrippen uit de doeken gedaan.

Invoering van het computing-curriculum

Hoe werkte de invoering van het nieuwe curriculum in de praktijk? Uit de antwoorden op onze vragenlijsten konden we het volgende afleiden:

- over het algemeen geven de Britse basisscholen 1 uur per week les in computing aan alle klassen, van het primair onderwijs;
- op dit moment worden de lessen nog vaak gegeven door externe computing-docenten. Zij zullen hun collega-docenten in de komende jaren bijspijkeren, tot die genoeg kennis en ervaring hebben opgedaan om het stokje over te nemen;
- er is een algemeen actieplan opgesteld, om de invoering van het curriculum gestructureerd te realiseren;
- ruim de helft (60%) van de scholen had daarnaast een gedetailleerd stappenplan uitgewerkt, waarin precies stond uitgestippeld wat er in de komende jaren moest gebeuren om het curriculum zo goed mogelijk te implementeren;
- de implementatie van het nieuwe curriculum bleek in de praktijk niet zo heel erg lastig, omdat de scholen veelal gebruik maakten van: gespecialiseerde computing-docenten die al langer bij de (grotere scholen) werkten, docenten die via CAS waren opgeleid in computing, of docenten die ze via CAS hadden aangetrokken;
- de meeste (reguliere) docenten vonden de stap om zelf computing-lessen te geven te hoog gegrepen. Maar gedurende het schooljaar (2014-2015) raakten ze enthousiaster en minder sceptisch, en kregen ze meer zelfvertrouwen, vooral in de onderbouw. De meeste docenten volgen bijscholingscursussen (o.a via CAS), en maken zich op om op termijn zelf de lessen over te kunnen nemen;
- vooral oudere docenten vonden het nieuwe computing-curriculum behoorlijk lastig. Voor deze groep bleek het over het algemeen zeer moeilijk om computing-lessen te geven;
- de leerlingen reageerden positief. Dat bleek niet alleen uit de gretigheid waarmee ze het nieuwe vak omarmden, maar ook uit het gebruik van de buitenschoolse computer- en code-clubs die op bijna alle basisscholen (80%) aanwezig zijn, en veelal zijn overtekend;
- scholen hebben bijna allemaal een apart computing-lokaal ingericht, veelal met steun van organisaties (zoals CAS) en bedrijven;
- de leerlijnen zijn vrijwel allemaal gedefinieerd op basis van het boek *Computing in the national curriculum. A guide for primary teachers* van Miles Berry, dat onder de vlag van CAS en Naace is uitgegeven.[16]

Veel Britse basisscholen, zo blijkt uit de ingevulde vragenlijsten, hebben een speciale computing-docent aangesteld die als coördinator fungeert. Met als taken:

- het opstellen en implementeren van een plan van aanpak;
- het instrueren en bijstaan van collega-docenten;
- soms: een computerruimte inrichten;
- en soms: een computerclub/codeclub (of andere buitenschoolse activiteit) organiseren.

Vaak zijn deze coördinatoren tijdelijk in dienst (gedurende 2 tot 4 jaar).

Casus 1: Killigrew Primary School

James Large is computing-coördinator van *Killigrew Primary School* in St. Albans, een stadje ten noorden van Londen. Er wordt – zoals op bijna alle basisscholen – één uur per week lesgegeven in het nieuwe vak computing.

"De veranderingen zijn groot," zegt Large. "Voorheen werd er vooral met Microsoft-programma's als Powerpoint, Word en Excel gewerkt, maar nu bouwen we Lego-robots, en gebruiken we Scratch om games te bouwen en interactieve verhalen te maken. En nog veel, veel meer."

De school heeft een apart computerlokaal met 30 computers, zodat de leerlingen ook individueel taken kunnen uitvoeren. "Hier werken we met Lego WeDo. Ook Scratch en Python zijn geïnstalleerd," vertelt Large.

Large is al acht jaar docent, waarvan drie jaar als computing-specialist; een titel die hij via CAS-cursussen en -examens verkreeg. Hij heeft het computing-curriculum uitgewerkt en geïmplementeerd, spijkert zijn mededocenten bij, en helpt ze bij de invoering van het nieuwe curriculum. Bij elke les van Large is de bijbehorende klasse-docent aanwezig, om te leren van zijn kennis en aanpak. Volgend jaar zullen de klasse-docenten deze lessen zelf geven, met enige bijstand van Large. "Sommige oudere docenten zijn er wat huiverig voor," zegt Large. "Maar anderen pikken het heel snel op. Een van hen kocht *Computer Coding for Children*, uitgegeven door Dorling-Kindersley. Hij geeft nu vol zelfvertrouwen programmeer-lessen. Daarna kocht de school 100 exemplaren van het boek."

In de laagste klassen van Killigrew wordt computing vaak gecombineerd met andere lessen. Men had bijvoorbeeld al een les in navigatie, die goed bleek te kunnen worden gebruikt bij het nieuwe computing-curriculum. Leerlingen moeten hun partner over het schoolplein zien te navigeren, gebruikmakend van commando's als vooruit, achteruit, links, rechts en

stop. Dat was al een programma-onderdeel, maar nu heeft de school het aangepast aan de computing-lessen, door de les te koppelen aan begrippen als 'algoritme', 'instructie' en 'iteratie' (herhaling).

Afbeelding 2 - Navigatielessen (Kiligrew Primary School, St.Albans)

Large: "Kinderen vinden het geweldig. Ze leren hoe ze iets kunnen opsporen en onderzoeken via het internet, en hoe ze met elkaar kunnen communiceren via e-mail en blogs." Veiligheid is daarbij volgens Large een van de belangrijkste zorgen. "Ik gebruik *360 degree safe* om de online-veiligheid van deze school te monitoren. Dat heeft ons erg goed geholpen."[17]

De school biedt ook buitenschoolse activiteiten aan. Op vrijdag tijdens lunchtijd is er bijvoorbeeld een computer- annex codeclub. Die is zo populair dat er een wachtlijst moest worden ingesteld.

Verder heeft de school 'digitale leiders' aangesteld (één per klas), die aangesproken mogen worden door de andere leerlingen, en hun kennis over computers en software zoveel mogelijk met hen moeten delen. Deze whizzkids worden hiervoor niet gecompenseerd, maar krijgen wel status, en raken hierdoor hun imago van sneue nerds een beetje kwijt.

Casus 2: Malvern Wells Primary School

Ook bij *Malvern Wells Primary School* in Worcestershire hebben ze 'digitale leiders', maar dan met een heel ander doel. Namelijk: om de docenten bij te staan. Deze *digital leaders* komen uit de hoogste klassen van het primair onderwijs, en worden door computing-docent Matt Warne ingewijd in het curriculum. Anders dan bij Killigrew (zie Casus 1) geven de docenten van Malvern Wells namelijk zelf de computing-lessen in sleutelfase 1 (5-7 jaar).

In sleutelfase 2 (7-11 jaar) is de stof nog te ingewikkeld, en geeft Warne zelf de lessen. Onder andere in het gebruik van de computertalen Scratch, Logo en Python. Warne: "Ook doen we veel

aan bloggen, onder andere omdat we daardoor iets kunnen vertellen over online-veiligheid."

"In de toekomst," zo vertelt Warne, "zullen de docenten zelf computertalen moeten beheersen," een taak waarvoor Warne zich de komende jaren gesteld ziet. Het grootste probleem dat hij nu ziet, is dat sommige docenten niet de kennis in huis hebben om de snellere, slimme leerlingen te kunnen ondersteunen. Vandaar zijn keuze om de digitale leiders te trainen, en zelf nog de lessen in de bovenbouw te geven, totdat de docenten genoeg kennis en ervaring hebben opgedaan om zelf de lessen te kunnen verzorgen.
"De reactie van de leerlingen was overweldigend," aldus Warne. "Ze vinden het geweldig om uitgedaagd te worden. Ook ontdekten we dat je enorm veel bruggen kunt slaan met rekenen en wiskunde. Veel meer dan we ons in eerste instantie gerealiseerd hadden."

Casus 3: Wybourn Community Primary School

Julian Wood van *Wybourn Community Primary School* in Sheffield is CAS-teacher en computing-coördinator. Hij geeft zo'n 10 uur computing-les per week waarbij ook zijn collega-docenten actief deelnemen, "zodat ze er ook iets van leren."
Toen het nieuwe computing-curriculum van start ging, was een aantal docenten namelijk erg bezorgd over hun computerkennis en programmeervaardigheden. Wood gaf het afgelopen jaar twee keer een tweedaagse training aan de docenten. "Sommigen zijn nog steeds onzeker, maar anderen kunnen inmiddels al behoorlijk overweg met Scratch."

Wood geeft niet alleen training aan docenten van zijn eigen school, maar ook aan docenten van andere scholen in de nabije omgeving.
"Ik vind het zelf ook prettig om contact met andere computing-docenten te hebben. Zo kom je op ideeën. Verder is er heel wat materiaal beschikbaar op het internet." Als docent die verantwoordelijk is voor 'het creatieve gebruik van technologie' organiseert Wood jaarlijks verschillende activiteiten om het gebruik van technologie en computers te promoten. Zo doet de school mee aan de jaarlijkse (Europese) Safer Internet Day in februari (waar ook Nederland aan meedoet). "Dat soort activiteiten is belangrijk om de aandacht te vestigen op het belang van het nieuwe computing-vak," aldus Wood.

2.3 Computing in het secundair onderwijs

Ook het secundair onderwijs in het Verenigd Koninkrijk omvat twee key stages (sleutelfases):
- sleutelfase 3 voor 11- tot 14-jarigen;
- sleutelfase 4 voor 14- tot 16-jarigen.

Hieronder volgen eerst de ministeriële richtlijnen voor beide fases. Daarna hoe het werkt in de praktijk.

Sleutelfase 4 is minder specifiek uitgewerkt dan de voorgaande fasen. De reden daarvoor is dat er veel afhangt van het soort specialisatie dat de school aanbiedt.

Sleutelfase 3 (11- 14-jaar)

- computationele abstracties ontwerpen, gebruiken en evalueren, om realistische problemen en fysische systemen te kunnen modelleren;
- een aantal belangrijke algoritmen begrijpen die representatief zijn voor computational thinking. Bijvoorbeeld: algoritmen voor sorteren en algoritmen voor zoeken;
- verschillende algoritmen voor hetzelfde probleem kunnen vergelijken;
- twee of meer programmeertalen leren gebruiken, waarvan in ieder geval één tekst-gebaseerd is, om verschillende computationele vraagstukken op te kunnen lossen;
- datastructuren leren doorgronden, zoals arrays en tabellen;
- modulaire programma's ontwerpen, die procedures en functies bevatten;
- eenvoudige Booleaanse logica (met AND, OR en NOT) leren begrijpen, evenals een aantal toepassingen daarvan in circuits en programma's;
- begrijpen hoe het binaire stelsel werkt, en eenvoudige bewerkingen leren uitvoeren (zoals binair optellen, en converteren naar decimaal);
- inzicht krijgen in de hardware- en software-componenten die deel uitmaken van een computersysteem, en begrijpen hoe deze met elkaar en met andere systemen communiceren;
- begrijpen hoe instructies worden opgeslagen en uitgevoerd;
- begrijpen hoe verschillende soorten data (zoals tekst, beeld en geluid) gerepresenteerd en gemanipuleerd kunnen worden;
- creatieve projecten opzetten en uitvoeren, waarmee ambitieuze doelstellingen verwezenlijkt kunnen worden;
- de basisprincipes van object-georiënteerd programmeren leren begrijpen, met aandacht voor aspecten als hergebruik, betrouwbaarheid en gebruiksvriendelijkheid;
- begrip ontwikkelen voor veiligheid, respect en verantwoordelijkheid, inclusief de bescherming van identiteit en privacy;
- ongepaste inhoud en dubieuze contacten leren herkennen, en weten hoe veiligheidsproblemen gerapporteerd kunnen worden.

Sleutelfase 4 (14-16 jaar)

- talent-ontwikkeling, creativiteitsontwikkeling en kennisoverdracht, op de gebieden informatica, digitale media en informatietechnologie;

- het ontwikkelen van analytische, probleemoplossende en design-vaardigheden;
- leren begrijpen hoe technologische vernieuwingen de veiligheid (waaronder online identiteit en privacy) kunnen beïnvloeden, zowel positief als negatief;
- leren hoe je je zorgen en twijfels kunt identificeren en rapporteren.

Kern van het curriculum

In het secundair onderwijs worden de grondbeginselen van informatica verder uitgewerkt. Informatietechnologie wordt onderwezen als praktisch hulpmiddel, en er wordt vooral gewerkt aan *computational thinking*.

Er moet met verschillende programmeertalen worden gewerkt, en van de leerling wordt gevraagd om (zelfstandig of groepsgewijs) complexe computertoepassingen voor een specifiek publiek en een specifiek doel te bedenken en fouten op te sporen.

Ook wordt geleerd om binair te rekenen, om logica toe te passen, en om hardware en software te beheersen.

Sleutelfase 4 is vooral bedoeld om gemotiveerde leerlingen de kans te bieden een officieel diploma te behalen, waarmee ze verder kunnen op IT-gebied. In deze fase is computing ook een keuzevak (met meer contacturen als gevolg).

Invoering van het computing-curriculum

Hoe werkte de invoering van het nieuwe curriculum in de praktijk? Uit de beantwoording van onze vragenlijsten konden we het volgende afleiden:

- in sleutelfase 3 (11-14 jaar) wordt gemiddeld 2 uur per week computing-les gegeven;
- in sleutelfase 4 (14-16 jaar) wordt gemiddeld 3 tot 5 uur per week computing-les gegeven (met inbegrip van computing als keuzevak);
- tweederde van de scholen heeft een geheel nieuw plan van aanpak geschreven, waarin precies staat welke leerlijnen ze in de twee laatste sleutelfases zouden volgen;
- een derde van de scholen houdt de algemene lijnen en aanwijzingen van *Computing at School* aan, en probeert al doende nieuwe leerlijnen op te zetten. Ze willen eerst meer ervaring opdoen met het nieuwe vak, voordat ze een compleet lesprogramma uitwerken;
- over het algemeen (70%) wordt er vooral met gespecialiseerde docenten gewerkt, in ieder geval in sleutelfase 4. Veel van deze computing-docenten waren al werkzaam op de school, maar hadden minder lesuren, minder verantwoordelijkheid en over het algemeen een lagere status;

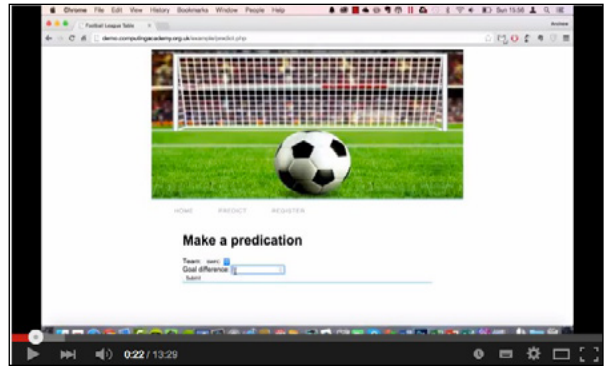
- in sleutelfase 3 worden de lessen regelmatig door wiskunde- of natuurkundedocenten gegeven, die vaak door CAS zijn bijgespijkerd of van nieuw materiaal zijn voorzien;
- het nieuwe curriculum leverde meestal (85%) geen grote weerstand binnen de school. De weerstand die er was, had vrijwel altijd te maken met docenten en directeuren die het computing-curriculum een modeverschijnsel vinden;
- de leerlingen zijn over het algemeen zeer enthousiast (net als de ouders), al vinden sommige leerlingen de stof behoorlijk pittig;
- in ruim 60% van de scholen is er een computer- of codeclub waar scholieren in de tussen- of avonduren kunnen leren omgaan met computers, software en programmeergereedschap (onder begeleiding).

Casus 1: Knutsford Academy

Op *Knutsford Academy* in het district Cheshire wordt door een staf van 60 docenten lesgegeven aan zo'n 1100 leerlingen in de leeftijd van 11 tot 18 jaar. Hoofd van het Computing Learning Centre is Andrew Clarke, die vlak nadat in 2012 het nieuwe curriculum werd aangekondigd, het nieuwe informaticaonderwijs heeft vormgegeven.[18]

Knutsford heeft een eigen curriculum ontwikkeld (zie Bijlage 4.4) op basis van de ministeriële vereisten en de algemene richtlijnen en aanwijzingen van CAS. "Het nieuwe nationale curriculum werd in 2012 aangekondigd, waarna we onze werkschema's hebben herschreven, zodat er veel meer informatica-concepten in werden verwerkt," zegt Clarke. "Het ministerie eist dat we *computing* geven, maar de scholen zijn verantwoordelijk voor hun eigen werkplannen en -schema's."

In de onderbouw (sleutelfase 3, lesjaar 7-9, leeftijd 11-14) geeft men nu sinds augustus 2014 één uur per week les in computing. Leerlingen van 11 en 12 jaar leren over online veiligheid, ze leren programmeren met Scratch en Small Basic, ze maken stroomdiagrammen, ze programmeren verkeerslichten, ze leren robots besturen met Flowol, en maken webpagina's in HTML.



Afbeelding 3: Video-les van het eigen YouTube-kanaal van Knutsford Academy, over gebruikers-input in PHP

Leerlingen van 13 jaar en ouder leren op Knutsford programmeren in Python en PHP. Daarnaast leren ze databases bouwen met MySQL, en krijgen ze les in logische poorten (logic gates) en waarheidstabellen. Verder krijgen ze onderricht in het binaire getallensysteem, inclusief optellen en aftrekken.

Vanaf lesjaar 10 (leeftijd 14-15) wordt aan de Knutsford Academy in sleutelfase 4 het (facultatieve) hoofdvak GCSE-computing aangeboden voor hen die graag met dit vak verder willen (zowel voor informatica als voor ict). De bijbehorende studielast is 2 tot 3 uur per week. Ook bestaat de mogelijkheid om A-level informatica te volgen voor leerlingen die Computer Science willen gaan studeren op de universiteit. De bijbehorende studielast is 4 tot 5 uur per week.

Bij GCSE-computing krijgen de leerlingen dezelfde stof als eerder genoemd, maar dan op een hoger niveau. Daarnaast studeren ze volgens Clarke ook de Software Development Lifecycle, hexadecimale getallen, hoe geluid en beelden binair worden gerepresenteerd, en leren ze over de structuur van netwerken (voornamelijk internet, incl. IP-adressering en de belangrijkste protocollen).

Wat Clarke opviel in het eerste jaar dat er computing werd gegeven (2014-2015), was dat leerlingen behoorlijk positief en enthousiast zijn. Maar soms gaat het mis bij leerlingen die zich dreigen te vertillen aan de stof. “Daarom is het ook zo belangrijk om de lessen te differentiëren en verschillende instapniveaus in te bouwen,” aldus Clarke. Hij heeft daarvoor video’s (in een eigen YouTube-kanaal) en papieren instructies gemaakt, waarmee leerlingen individueel en op hun eigen tempo kunnen werken.

Casus 2: Kent Secondary School

Niet bij elke school ging het zo voortvarend als op Knutsford (Casus 1). Michael Jones van *Kent Secondary School* in Kent vertelt dat er niet echt een plan van aanpak was gemaakt, maar dat ze probeerden het oude ict-curriculum langzaam om te buigen naar het nieuwe computing-curriculum. Dat deden ze in eerste instantie met de jongere leerlingen (vanaf 11 jaar). “Dat leek ons het beste om mee te beginnen.”

Een van de onderdelen die werden ‘omgebogen’, waren de Photoshop-lessen. Die bleken namelijk de mogelijkheid te bieden om de bestaande praktijk-instructies uit te breiden met complexere informatica- en IT-inzichten, zoals datarepresentatie, bit-diepte, compressie, en het hexadecimale schema voor kleurcodes.

Later werd er iets gestructureerder gewerkt, onder andere met hulp van CAS. Ook kwam er een plan voor het verdiepende GCSE-onderwijs in sleutelfase 4 (vanaf 14 jaar).

"Niet iedereen was even enthousiast," vertelt Jones. "Vooral de oudere docenten twijfelden nogal over hun eigen capaciteiten, waaronder de mogelijkheden om zich de nieuwe stof eigen te maken, en te kunnen geven. "Een derde van de docenten was enthousiast en ging aan het werk. Ook een derde was wel enthousiast, maar moest flink ondersteund worden. En tenslotte waren er de struisvogels die hun hoofd in het zand staken en dachten dat het allemaal wel voorbij zou gaan."

Het belangrijkste – voor de docenten – is volgens Jones niet het programmeren, maar begrijpen wat *computational thinking* nu precies inhoudt. "De uitdaging was om de nieuwe ideeën te doorgronden. Voor sommige docenten was dat te hoog gegrepen, waardoor ze teleurgesteld werden."

Volgens Jones kwam dit onder andere door het niveauverschil tussen de docenten: niet alleen op hun eigen vakgebied, maar vooral wat het digitale domein betreft. "We hebben hier nu twee gespecialiseerde informatica-docenten met universitaire diploma's in dienst," vertelt Jones. "Maar er zijn ook docenten die jarenlang ict-lessen gaven op basis van Microsoft Office. Bij complexere informatica- en programmeerconcepten hebben die een probleem."

Casus 3: Dereham Neatherd Highschool

Dereham Neatherd Highschool in Dereham koos een langere en gestructureerdere route. Op deze school (1500 leerlingen, 75 docenten) wordt al vanaf 2010 in de bovenbouw (sleutelfase 4) lesgegeven in computing (GCSE), terwijl voor de onderbouw (11-14 jaar), drie jaar voordat het verplicht werd, een nieuwe curriculum werd ingevoerd. "We zijn hier erg vooruitstrevend," verklaart Adam Gibson, hoofd informatica van de school. "We waren een beetje vroeg met onze computing-lessen, maar dat betaalt zich nu terug."

De algemene lessen voor 11-14 jarigen worden door niet-specialisten gegeven, zoals wis- en natuurkundedocenten. Voor de GCSE-lessen van de oudere leerlingen wordt wel gebruik gemaakt van specialisten, waaronder Gibson zelf. Elke week geeft Gibson tijdens lunchtijd bijles aan de docenten, zodat ze de kans hebben om hun kennis van programmeertalen en informatica-grondbeginselen te vergroten. Binnenkort wordt een compleet computer-lokaal met 60 computers geopend.

Casus 4: Fairfield High School for Girls

Op *Fairfield High School for Girls* in Manchester, een middelgrote school met 900 leerlingen en 80 docenten, is wiskundedocent Emily Nisbet verantwoordelijk voor de implementatie van het nieuwe curriculum. In de eerste klas wordt lesgegeven over de beginselen van de computer, het binaire stelsel, en algoritmes. Er wordt geprogrammeerd in Scratch (om programmeerproblemen op te lossen, en om onderwerpen als 'reeksen' en 'herhaling' te demonstreren). Daarnaast worden de principes van machinecode en de Centrale Verwerkingseenheid (CPU) uitgelegd, maar ook hoe beelden worden opgeslagen. Daarnaast komt het internet (waaronder HTML en het web) aan de orde.

In de tweede klas wordt het programma op een hoger niveau gebracht, en gaat het ook over de interactie met de gebruiker. In de derde klas is er meer aandacht voor geluidstoepassingen en *computational thinking*.

"Toen de plannen bekend werden, heb ik een actieplan uitgewerkt, dat ik steeds verder heb verfijnd," vertelt Nisbet. "Ik was al begonnen met de GCSE-lessen voor sleutelfase 4, zodat we ons daaraan konden optrekken. Via CAS en online lessen van de universiteit van Manchester heb ik de leemtes in mijn kennis kunnen opvullen. Daarna heb ik nieuwe lesplannen uitgewerkt en ben ik de staf hier gaan trainen. We komen om de 14 dagen bij elkaar en ontwikkelen het lesmateriaal gezamenlijk."

Nisbet schat dat je zo'n drie jaar nodig hebt om alle elementen van het nieuwe computing-onderwijs te implementeren. "We hebben het hier in hoog tempo aangepakt, maar de hoeveelheid kennis die je moet zien te verwerven om in het voortgezet onderwijs computing-lessen te geven, is aanzienlijk. Daar moet je je niet in vergissen."

2.4 Didactisch model en lespraktijk

Hieronder volgen de didactische uitgangspunten van Groot-Brittannië. Men baseert zich daarbij op CAS in het algemeen en Miles Berry in het bijzonder. (Zie onder 'Berry' in de Bibliografie.)

Didactiek op basis van maken en doen

In de handleiding van CAS, die massaal wordt gebruikt, staat het doen voorop. Eerst iets maken, vervolgens de leerlingen over hun eigen werk laten reflecteren, en pas daarna de theorie uitleggen; dat is de kern van het didactisch model. Alleen bij hoge uitzondering worden klassieke theorielessen gegeven; voor de rest zijn de computing-lessen bijna altijd projectlessen,

waarbij leerlingen in groepjes of zelfstandig iets moeten doen of maken (in veel gevallen iets programmeren), waarna de docent aan het eind van de les de resultaten bespreekt en vervolgens de achterliggende principes uitlegt en verwijst naar het materiaal in boeken en op websites.

Adviezen van CAS

Volgens CAS wijzen onderzoek en ervaring het volgende uit:

- je leert de beginselen van informatica beter als je programma's schrijft;
- je wordt digitaal geletterder als je moet *documenteren* wat je geleerd hebt, bijvoorbeeld door middel van een handleiding, een blog, een geluidsopname of een *screencast*;
- je leert informatietechnologie beter te gebruiken als je er iets creatiefs mee moet doen, zoals een presentatie geven, een website maken of een video tonen. De didactische waarde en de inzet worden merkbaar hoger als leerlingen hun werk aan anderen moeten tonen;
- computing-kennis verbetert sterk als je theoretische vragen kunt stellen aan gevorderde medeleerlingen ('digitale leiders').

Adviezen van Miles Berry

Miles Berry is een van de leidende personen binnen CAS. Hij adviseert:

Projectmatig werken – "Probeer zoveel mogelijk praktische, creatieve projecten te verzinnen waar de leerlingen aan kunnen werken – individueel, samen met een partner, of in een groepje," adviseert Miles Berry in zijn boek *Computing in the National Curriculum – A Guide for Primary Teachers* (2013). "Zo werkt het namelijk ook in de echte wereld en op de universiteiten."

Koppelen aan belevingswereld – Projecten leveren volgens hem het meeste op als je ze kunt koppelen aan de belangstellingen en voorkeuren van de leerlingen. "Dat kunnen andere terreinen van het curriculum zijn, het schoolleven of interessegebieden buiten school."[19]

Presenteren en delen – Berry wijst vooral op het nut van een publiek met dwingende ogen. "Dat kan bijvoorbeeld door resultaten te presenteren aan elkaar, schrijven voor een openbaar weblog, het maken van software of content voor jongere leerlingen, of Scratch-creaties delen met anderen (op scratch.mit.edu)."

Valkuilen vermijden – Berry signaleert twee mogelijke problemen:

- valkuil: het open-einde karakter van veel projecten.
- bij samenwerken: hoe beoordeel je de individuele bijdragen?

Wat het eerste punt betreft wijst hij op het belang van afbakening: elke les moet over één onderdeel of aspect gaan, dat ook telkens afgerond moet worden. Het kan bijvoorbeeld gaan om: het maken van een subroutine, het verbeteren van een eerder project(deel), het nakijken van een (deel)project van iemand anders, of het produceren van documentatie bij een eerder gemaakt programma. Zolang het maar iets is wat in zijn geheel kan worden afgerond in één les.

En wat het tweede punt betreft: ook op de individuele bijdrage van de leerling moet de docent beducht zijn. Elke leerling moet een bepaalde rol in een groepsproces krijgen, en op het volbrengen van die specifieke rol moet hij beoordeeld worden.

Dat laatste betekent wel dat er veel tijd besteed zal moeten worden aan:

- het voeren van discussies over wat de beste, slimste, mooiste weg is om tot een oplossing te komen;
- het verfijnen van programma's, zowel theoretisch als praktisch;
- het opsporen van fouten (debugging).

Tip: debuggen is ook te onderwijzen door als docent opzettelijk fouten in een goede, schone listing te maken, en de leerlingen die fouten te laten vinden en herstellen.

Lesvormen zonder beeldscherm

Betekent het Engelse computing-curriculum dat de leerlingen altijd achter een scherm zitten te coderen en te programmeren? Niet altijd en zeker niet voor de jongste leeftijden (sleutelfase 1), zo benadrukt CAS. Veel van de theoretische beginselen van informatica, informatietechnologie en digitale geletterdheid zijn heel goed uit te leggen met behulp van rollenspellen, met pen en papier of door middel van creatief schrijven.

Op CS Unplugged (www.csunplugged.org) en CS4FN (www.cs4fn.org) zijn inmiddels vele voorbeelden verzameld van informaticalessen waarbij geen enkele computer gebruikt hoeft te worden. Een groot deel daarvan is inmiddels vertaald in het Nederlands.

Ook voor oudere leerlingen (sleutelfase 2 en hoger) hoeft niet altijd een beeldscherm gebruikt te worden: het ontwerpen van een systeem of een programma, de routering en de achterliggende ideeën, kunnen allemaal op papier worden uitgevoerd. En discussies over een bepaalde aanpak of oplossing kunnen volgens CAS het beste *unplugged* gevoerd worden. Beeldschermen leiden dan alleen maar af.

2.5 Computing-coördinatoren

Goed werkend computing-onderwijs staat of valt met de aanstelling van een centrale computing-coördinator. Hij (m/v):

- is hoofd (of team-leider, etc.) van het computing-onderwijs van de school;
- heeft vaak informatica gestudeerd (of was ooit werkzaam in de IT-sector);
- is soms al werkzaam als docent in het primair onderwijs, of wiskunde/natuurkunde-docent in het secundair onderwijs;
- is bijna altijd geaccrediteerd door CAS.

Taken en verantwoordelijkheden

De (Britse) computing-coördinator is verantwoordelijk voor:

- het opstellen van het computing-curriculum;
- lesgeven in de computing-vakken;
- het opleiden en ondersteunen van collega's.

Daarnaast is hij ook de schakel tussen de school en CAS, en is hij vaak ook regionaal actief bij het uitwisselen van ervaringen en lessen (via de zogenaamde CAS-hubs; zie 2.6).

De computing-coördinator wordt door het schoolbestuur aangesteld of als zodanig aangemerkt, en is dus de aanjager van het computing-onderwijs. Hij maakt, in samenwerking met het schoolbestuur, een lange-termijnplan en werkt vandaaruit, veelal met bestaand CAS-materiaal, een lesplan uit, vaak samen met collega's. Vaak organiseert hij ook computer- of codeclubs na schooltijd, en is hij het aanspreekpunt voor docenten, ouders en schoolbestuur.

Beloning en aanzien

De verantwoordelijkheid die de computing-coördinator draagt is groot, en daarom wordt hij vaak extra beloond: financieel en/of met extra faciliteiten. Extra beloningen – materieel of immaterieel – hoeven niet alleen vanuit de school te komen, maar kunnen ook van buiten komen. Bijvoorbeeld van sponsors of via de CAS-organisatie.

Daarnaast is het coördinatorschap sowieso status-verhogend. 'Master-teachers' in CAS-stijl genieten inmiddels veel aanzien omdat ze in korte tijd het nieuwe computing-curriculum wisten te implementeren en zo ervoor zorgden dat de ministeriële verplichtingen op tijd nagekomen konden worden.

2.6 Computing-docenten

Wat moet een computing-docent precies kunnen? Voorheen was het voldoende om enige globale kennis te hebben van computers, internet, en

kantoor-software. Maar het nieuwe curriculum stelt heel andere eisen. Nu moet een docent die computing geeft opeens van alles weten over algoritmes, procedures, debugging, protocollen en programmeertalen. Om nog maar te zwijgen van de didactiek om al die kennis over te dragen.

De vereiste competenties verschillen van school tot school, omdat scholen zelf hun curriculum kunnen invullen. Om een indruk te krijgen van wat er zoal gevraagd wordt, kun je vacatures bekijken. Zie Bijlage 4.12 – Vereisten voor een *computing-docent* voor een voorbeeld.

Bijscholing

Een van de grootste problemen van het nieuwe Britse computing-curriculum was dat veel docenten moesten worden bijgeschoold, inderhaast 'ingevlogen', of losgeweekt uit hun IT-baan.[20] De verschillen tussen het oude en het nieuwe curriculum zijn immers aanzienlijk.

Hoe konden ze dat in korte tijd voor elkaar krijgen? CAS definieerde vier routes voor bijscholing:

Route 1 – via de Computer Science-faculteiten. Op de Britse universiteiten zijn speciale bijspijkercursussen opgezet voor de beste docenten (de zogenaamde 'Master Teachers').

Route 2 – via Master Teachers (zie boven). Die kunnen op hun beurt weer docenten op hun eigen school of nabijgelegen scholen bijspijkeren. Daarnaast helpen ze bij het invullen van het curriculum, het bedenken van lesprojecten, en het kiezen van hardware en software. Dit verloopt grotendeels via *hubs* (zie onder): lokale of regionale clubs van computing-docenten.

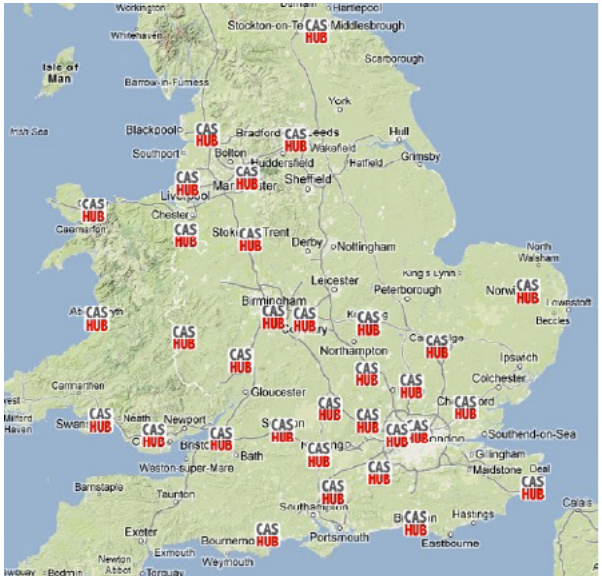
Route 3 – via IT-specialisten uit de software-industrie. CAS heeft zich er sterk voor gemaakt dat scholen en docenten worden ondersteund door IT-specialisten, zoals programmeurs en systeembeheerders. Die kunnen hun kennis en ervaring onder andere delen via de *hubs*. Een deel van deze specialisten heeft ook een tijdelijke aanstelling gekregen op de scholen.

Route 4 – via computer- en codeclubs. Deze centra voor buitenschoolse computing-activiteiten zijn niet alleen bedoeld voor de leerlingen maar houden zich ook bezig met het bijspijkeren van docenten[21].

CAS-hubs

Om het gebrek aan computing-kennis bij de docenten te compenseren heeft CAS het *Network of Excellence* opgericht: een virtuele gemeenschap van 'Master Teachers', computerexperts en IT-deskundigen die in lokale en regionale *CAS-hubs*

docenten proberen te enthousiasmeren, te helpen en op niveau te brengen.



Afbeelding 4: Lokale en regionale CAS-hubs

Deze hubs (netwerk-centra) worden geleid door vrijwilligers uit de CAS-gemeenschap, en ondersteunen scholen en docenten uit één regio. De activiteiten verschillen van hub tot hub, en hangen af van de coördinator en de leden van de gemeenschap. Maar vaak gaat het om bijeenkomsten die eens per week, per maand of per kwartaal worden georganiseerd, vaak in de school van een van de hub-leden. Op deze bijeenkomsten wordt kennis uitgewisseld, vaak door een lesvoorbeeld, een lesmiddel of een lespraktijk te demonstreren, meestal met discussie na afloop. Soms is er een externe spreker van de universiteit of uit het bedrijfsleven.

2.7 Toetsing en examinering

Hoe meten de Britten of een leerling voldoende vorderingen maakt, de stof begrijpt, en een eventueel volgend niveau aankan?

- in het primair onderwijs zijn er geen landelijke toetsen (zoals bij ons de Cito-toetsen) en zeker niet op het gebied van computing;
- in het secundair onderwijs is er alleen in het allerlaatste stadium van de bovenbouw (sleutelfase 4) een examen, als leerlingen het vak computing op GCSE- of A-level hebben gekozen, om te bepalen of ze voldoen aan de eisen van een vervolgopleiding.

Het toetsen van de voortgang van de leerlingen ligt dus geheel bij de docent.

Beoordelingstechnieken

Volgens de CAS-docentenhandleiding kun je het beste uitgaan van 'Assessment for Learning'-technieken (AfL), zoals *self-assessment*, *peer-assessment* en *machine-assessment*, ondersteund door technieken als *target setting* en open vragen stellen.[22]

Omdat in het computing-onderwijs de nadruk ligt op het *maken* van programma's, projecten en systemen, kun je de desbetreffende resultaten heel goed laten beoordelen door medeleerlingen. Je kunt gezamenlijk discussiëren of een programma of een oplossing al dan niet goed werkt, en wat er eventueel aan te verbeteren is. Dit is ook dagelijkse praktijk in de IT-wereld. In de 'echte' programmeurswereld zijn *code reviews* en *pair programming* heel gebruikelijk.

Handige hulpmiddelen

Het lastigste, volgens CAS, is om te meten in hoeverre leerlingen begrijpen wat ze hebben gedaan. Veel beginselen en theoretische grondslagen worden immers pas na of tijdens het maakproces uit de doeken gedaan, met het risico dat ze ondergesneeuwd kunnen raken in de euforie van de praktijk. Hoe los je dat op? Hieronder enkele suggesties uit de praktijk.

Blogs – "De effectiefste manier om te kijken wat iedereen heeft geleerd, is het starten van een klassenblog," stelt Mike Berry in de CPD-Toolkit voor docenten.[23] "Vraag aan de leerlingen om hun werk daarop te zetten, en de computationele denkprocessen die erachter schuilen, te documenteren. Laat ze vooral vertellen over de moeilijkheden die ze moesten overwinnen."

Volgens Berry werken dit soort blogs vooral goed als ze commentaar en reacties oproepen van andere leerlingen en docenten. En: "als je het goed doet, kun je met behulp van *tags* een systeem opzetten waardoor het mogelijk is de voortgang van kennis en competenties te volgen."

Interviews – de CAS-docentenhandleiding raadt ook aan om leerlingen aan het eind van een project te interviewen over hun (computationele) denkproces tijdens het werken aan het project. Stel daarbij zoveel mogelijk open vragen zoals, Bijvoorbeeld: "Waarom koos je ervoor om dit op deze manier op te lossen?" Of: "Kun je me uitleggen hoe dit precies werkt?"

Eindexamen

Zoals gezegd is de enige landelijke test het examen aan het eind van sleutelfase 4, als de leerlingen die 'computing' als keuzevak hebben gekozen, hun GCSE- en Level-A-diploma's proberen te halen.

Ter voorbereiding op dat examen gebruiken docenten steeds vaker computerprogramma's (software-assessment) om het niveau van hun leerlingen in te kunnen schatten en waar mogelijk te verbeteren, zo bleek uit de beantwoording van onze vragenlijsten. Deze vorm van beoordeling

(software-assessment) wordt inmiddels ook meer en meer vóór sleutelfase 4 gebruikt om het niveau van de leerlingen te bepalen.

2.8 Infrastructuur

Hoewel veel lessen – zeker in sleutelfase 1 – zonder computer gegeven kunnen worden, zijn de meeste de projecten onuitvoerbaar zonder een goede infrastructuur, bestaande uit hardware, software, en internet-faciliteiten.

CAS heeft daartoe een document opgesteld waarin alle opties uitvoerig worden besproken.[24] Voor de onderstaande suggesties maakten wij tevens gebruik van de CAS-gidsen voor primary en secondary teachers.

Hardware

Voor sleutelfase 1 wordt aangeraden om te werken met programmeerbare speelgoed-robotjes, zoals Bee-Bot, Roamer, of Pro-Bots.

Voor sleutelfase 2 wordt aangeraden om sensoren, lampjes, motortjes, en goedkope Arduino-computers (formaat credit-card) aan te schaffen, waarmee werkende projecten gerealiseerd kunnen worden. Daarnaast kunnen in deze fase gebruikt worden: de Raspberry Pi (vergelijkbaar met de Arduino maar dan voor algemener gebruik), smartphones, en tablets. Voor de sleutelfases 3 en 4 is het handig om oude computers te hebben die door de leerlingen uit elkaar gehaald (en eventueel gerepareerd) kunnen worden. Microfoons en digitale camera's kunnen van pas komen bij het bouwen en aansturen van real-life-projecten, zoals een bewakingscamera. (In de laatste versie van Scratch kun je ook webcams en camera's aansturen). Ook de eigen smartphones en/of tablets van de leerlingen kunnen worden gebruikt.

Software

Voor sleutelfase 1 kán de programmeer-omgeving Scratch worden gebruikt, maar de voorkeur gaat uit naar programmeerbare robotjes (zie boven). Daar komt verder geen aparte software aan te pas.

Voor sleutelfase 2 worden onder andere Flow-Go en WeDo van Lego aangeraden.

Voor de sleutelfases 3 en 4 zijn de software-mogelijkheden zeer talrijk. Denk in ieder geval aan de programmeertalen Python en PHP, en aan het database-managementsysteem MySQL. Daarnaast worden schoolblogsystemen aangeraden, evenals Google Drive om samen te werken aan projecten.

Internet-verbinding

Dat een goede internet-verbinding onmisbaar is, behoeft geen betoog. Vergeet niet om voldoende aandacht te besteden aan de online-veiligheid van de leerlingen, en bespreek met ze hoe ze daar ook zelf aan kunnen bijdragen.

3. Wat kunnen Nederlandse scholen leren van de Britse ervaringen?

Het eerste computing-jaar (2014-2015) in Groot-Brittannië zit er inmiddels op. Dat brengt ons op de centrale vraag welke lessen we kunnen trekken uit het Britse computing-onderwijs, zowel op theoretisch (didactisch), als op praktisch (organisatorisch) gebied.

Het Britse onderwijs in het algemeen, en het computing-onderwijs in het bijzonder, zitten anders in elkaar dan wat we in Nederland gewend zijn. Nederlandse scholen zetten meestal in op ict-basisvaardigheden, computational skills, informatievaardigheden en mediawijsheid, een cluster dat ook wel aangeduid wordt met 'digitale geletterdheid'. De vraag is dus vooral wat scholen die verder willen met de eerste twee vaardigheden – ict-basisvaardigheden en computational skills – van de Britten kunnen leren. Daarover gaat dit hoofdstuk.

Een tweede verschil tussen Groot Brittannië en Nederland is dat we in Nederland geen nationaal curriculum kennen. De onderstaande lessen hebben dus geen betrekking op Nederland als geheel (op landelijk niveau) maar zijn bedoeld voor afzonderlijke scholen (in het bijzonder: schoolbesturen, schoolleiders, coördinatoren en docenten).

Opmerking vooraf: wie wil beginnen met computing-onderwijs, moet ruimte creëren in het curriculum. Die ruimte is er in ieder geval in het basisonderwijs, bijvoorbeeld in nieuwe vak 'wetenschap en techniek' (geplande invoering in 2020). In de onderbouw van het voortgezet onderwijs kan de vrije ruimte benut worden (al zijn daar natuurlijk meer gegadigden voor). Leg contact met andere scholen om te kijken hoe die dit opgelost hebben.

3.1 Zorg voor een solide fundering

Wat opvalt aan de Britse aanpak, is dat er eerst een stevig theoretisch fundament is gelegd. Van daaruit zijn er lessen en lesmaterialen ontwikkeld. Het advies is dus om eerst een goed fundament te leggen (en dat te delen met het team), om dat vervolgens uit te werken tot bijvoorbeeld een leerlijn. De principes en uitgangspunten die de Britten hebben geformuleerd (zie hoofdstuk 2 en de CAS-website), zijn ook bruikbaar in het Nederlandse onderwijs.

"Programmeren is de sleutel tot 21e-eeuwse vaardigheden" is de boodschap van Britse (maar ook Estse en Finse) onderwijskundigen en computing-ambassadeurs. Hoewel de bijbehorende claims (dat je door computing-onderwijs beter zou leren denken, en beter

opgewassen zou zijn tegen de eisen van deze tijd, etc.) niet gesteund worden door wetenschappelijk onderzoek, ligt het wel voor de hand om programmeren te zien als een middel om verschillende vaardigheden en inzichten rondom computertechnologie aan te leren.

Om aan uw team duidelijk te maken waar het over gaat, zou je gezamenlijk naar de inspirerende TEDx-lezing *Teaching creative computer science* kunnen kijken (beschikbaar via YouTube[25]).

Daarin wordt helder uitgelegd waar computing en programmeeronderwijs voor staan.

Vertaalslag

In Nederland wordt het fundament voor computing onderwijs gelegd door SLO. Zie: *Digitale geletterdheid en 21e eeuwse vaardigheden in het funderend onderwijs: een conceptueel kader* (SLO, 2014). Ook verschijnt er binnenkort (waarschijnlijk in januari 2016) een handig rapport, met als werktitel *Digitale geletterdheid, Computational thinking* (SLO, 2016).

3.2 Ontwikkel een visie

Computing- en programmeer-onderwijs stamp je niet zomaar uit de grond, getuige de Britse ervaringen. Het invoeren van een didactisch verantwoorde leerlijn invoeren kost – afhankelijk van ervaring en ondersteuning – zeker 3 en hoogstwaarschijnlijk meer dan 5 jaar[26]. Dat vereist een gedegen visie.

Dus: als je als school aan de slag wilt, vraag jezelf dan eerst af wat je precies wilt gaan onderwijzen en waarom. Alleen programmeren en computational thinking, of breder: digitale geletterdheid? In Groot-Brittannië is veel energie gestoken in een visie, en een kloppend verhaal waar iedereen achter kon staan.

Een mogelijk vertrekpunt is het werk van Seymour Papert. Hij ontwikkelde Logo, de eerste educatieve programmeertaal, en bouwt voort op de ideeën van Piaget. Wat hij toevoegt, is projectmatig werken. Paperts klassieker *Mindstorms, Children, Computers and Powerful ideas* (1980) is nog steeds een inspirerend en goed leesbaar boek, dat ook weer actueel is door de coding-discussie.

Bij meerdere scholen die onder één bestuur vallen, kan het overigens heel goed werken om eerst te experimenteren op één locatie, en daarvan te leren, zo blijkt. Vervolgens kun je dan een onderwijsplan maken dat integraal geïmplementeerd wordt. Ter oriëntatie kan het ook nuttig zijn om contact te zoeken met scholen die al ervaring hebben opgedaan. Zoals de

scholen die bij de PO-Raad een 'versnellingsvraag' hebben ingediend.[27]

Vertaalslag

Als je je als school wilt beperken tot ict-basisvaardigheden en computational thinking (in plaats van het – bredere – cluster 'digitale geletterdheid'), kunnen die onderwerpen ook prima ingebed worden in het onderwijsplan voor wetenschap en technologie. In het Techniekpact[28] is vastgelegd dat in 2020 alle basisscholen wetenschap en technologie structureel hebben ingevoerd in hun onderwijsprogramma.

Lees ter inspiratie: Onderwijs 2032 – Vindingrijk en nieuwsgierig van de PO-Raad en Platform Betatechniek.[29]

3.3 Benut de Britse didactische ervaringen

Wat de didactiek betreft, laten de ervaringen in Groot-Brittannië vier belangrijke dingen zien:

1. Begin bij de praktijk – Doen en maken moeten voorop staan, toont het Britse computing-onderwijs aan. Kennis opdoen over informatica, computing en computationeel denken is vele malen effectiever als er aan concrete projecten wordt gewerkt: een programma, een vraag, een oplossing voor een probleem, het opsporen van fouten, etc. Bovendien motiveert deze aanpak de leerlingen, en geeft het hen de mogelijkheid om resultaten te delen met anderen.

Richt de lessen dus zo in dat de grondbeginselen van computationeel denken uitgelegd en geleerd kunnen worden vanuit de praktijk. Eerst doen (met een kleine instructie vooraf, uiteraard), dan reflecteren, en dan pas de theorie uitleggen: dat is een van de kernlessen van het Britse computing-curriculum.

Vanzelfsprekend is deze aanpak (doen - reflecteren - uitleggen) niet heilig. Blijf nadenken en discussiëren over de beste didactiek. Probeer proefondervindelijk uit te vinden wat wel en niet werkt.

2. Werk groepsgewijs – Laat de leerlingen zoveel mogelijk in teams (subgroepjes) werken. Geef alle leerlingen een eigen welomschreven taak, en zorg dat ze hun werk documenteren, presenteren en uitleggen. Laat de rollen wisselen per project (teamleider, programmeur, debugger, documentalist, etc.)

3. Stimuleer discussies – Discussies tussen de leerlingen zijn essentieel, zo zeggen de Britse computing-docenten. Vragenrondes en

gesprekken over computing moeten een integraal onderdeel van het leerproces zijn. Bedenk daarbij dat er nooit één oplossing is, maar dat er altijd verschillende manieren zijn om tot een resultaat te komen (*quick & dirty*, elegant, zuinig, etc.), waarbij de ene oplossing niet per definitie beter is dan de andere.

4. Maak de lessen flexibel – Probeer bij elke les extra materiaal klaar te hebben liggen voor de snelle en de langzamere leerlingen. Deze methode wordt veel gebruikt om leerlingen meer vertrouwen te geven. Door bijvoorbeeld individuele online- of videolessen achter de hand te hebben, kun je minder snelle leerlingen de kans geven om zich te ontwikkelen, maar ook snelle leerlingen extra stof geven als ze hun taak al klaar hebben.

Daarnaast is het nuttig om lessen niet helemaal vol te plannen, zodat er ruimte blijft – voor de leerlingen – om te experimenteren. Dat vergroot de programmeervaardigheden. Ook blijkt het nuttig om de lessen op te delen in segmenten, zodat de leerlingen niet alleen maar aan het programmeren zijn, want dat wordt snel saai. Laat ze ook ontwerpen, documentatie maken, fouten opsporen, etc.

Vertaalslag

De bovenstaande principes zijn universeel voor het computing-onderwijs, en gelden dus ook voor de Nederlandse situatie. (Opmerking: slechts weinig docenten zullen echt goede programmeurs worden. Dat is geen enkel probleem. De kracht van een goede docent is zijn of haar kennis van didactiek en pedagogiek.)

3.4 Zorg dat er een gestructureerd lesplan ontwikkeld wordt

Laat de computing-coördinator een lesplan opstellen, bijvoorbeeld op basis van de leerlijn die op dit moment ontwikkeld wordt (door de PO-Raad en Kennisnet met hulp van SLO). Laat de computing-coördinator een stappenplan en een actieplan uitschrijven om computing-onderwijs op een realistische grondslag in te voeren. Volgens de Britse docenten die reageerden op onze enquête, is een periode van 3 tot 5 jaar nodig om een doorlopende, gestructureerde computing-leerlijn neer te zetten.

Inspiratie uit Groot-Brittanië kan onder andere ontleend worden aan:

- Berry, M., *Computing in the national curriculum. A guide for primary teachers.* CAS/Naace, 2013 (www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf)

- Berry, M., *QuickStart Computing; a CPD toolkit for primary teachers,* Computing at School, 2014 (www.quickstartcomputing.org)
- Berry, M., *QuickStart Computing; a CPD toolkit for secondary teachers,* Computing at School, 2014. (www.quickstartcomputing.org)
- (geen auteursvermelding), *The national curriculum in England - Key stages 1 and 2 - framework document,* Department for Education, sept. 2013 (www.gov.uk/government/uploads/system/uploads/attachment_data/file/335133/PRIMARY_national_curriculum_220714.pdf)
- (geen auteursvermelding), *The national curriculum in England - Key stages 1 and 2 - framework document,* Department for Education, dec. 2014 (www.gov.uk/government/uploads/system/uploads/attachment_data/file/381754/SECONDARY_national_curriculum.pdf)

Vertaalslag

Didactisch-inhoudelijk zou zo'n lesplan de Britse methode van 'al werkende weg' kunnen volgen:

- van concreet naar abstract (eerst iets maken, daarna de achterliggende theorie);
- van *unplugged* (zonder beeldscherm) naar *hands-on* (mét beeldscherm);
- leren door zelf ontdekken (met de docent als coach en de medeleerlingen als vraagbaak).

Inspiratie kan onder andere ontleend worden aan: Codekinderen, CodeUur en CodeStudio (zie Bijlage 4.5 – *Nederlandse initiatieven*).

3.5 Overweeg het inzetten van 'digitale leiders'

Gevorderde leerlingen kunnen zowel hun docenten als hun medeleerlingen ondersteunen. Ze kunnen bepaalde lessen – of delen daarvan – overnemen, en ze kunnen dienen als vraagbaak. CAS noemt dit 'digitale leiders' en hecht er veel waarde aan.

Voorbeelden van scholen die met digitale leiders werken, werden beschreven in Casus 1 en Casus 2 van paragraaf 2.2 - *Computing in het primair onderwijs.*

Vertaalslag

In Nederland moeten we misschien voorzichtig zijn met 'digitale leiders', vanwege onze egalitaire manier van denken. Maar dat is meer een kwestie van woordgebruik. Er is vanzelfsprekend niets op tegen om gebruik te maken van de kennis en ervaring van gevorderde leerlingen, en ze te laten helpen bij dingen die je als docent niet goed weet.

Een voorbeeld van zo'n leerling is Nick Jordan, die zijn eigen 'academy' runt. (Zie: www.jorcademy.nl) Er lopen meer Nick Jordans rond. Op het Berlage

Lyceum in Amsterdam werden bijvoorbeeld met succes *gamers* ingezet in het informatica-onderwijs, om te helpen bij het invullen van de lessen over het programma *Game Maker.*

Tip: organiseer regionale *coding-events* met docenten en leerlingen. Bijvoorbeeld in de vorm van een *edcamp*. (Een 'edcamp', ook wel 'unconference' genoemd, is een laagdrempelige, informele bijeenkomst, waarin de deelnemers ervaringen en ideeën uitwisselen.)

3.6 Stimuleer de interactie met andere vakken

Zoek samenwerking met andere vakken. Probeer uit te vinden welke lessen versterkt en verdiept kunnen worden met computing-aspecten. Denk bijvoorbeeld aan:

- taal- en spraaktechnologie in het taalonderwijs (zoals parsing, algoritmes voor woordafbreking, de basisprincipes achter *Google Translate*, etc.);
- sorteren en binaire getallen bij rekenen;
- routeplanners bij aardrijkskunde (hoe werkt *Google maps*?)
- beeldbewerking bij CKV.

Maak een tabel waarin duidelijk wordt welke computing-onderwerpen aansluiten bij de kerndoelen van andere vakken.

Vertaalslag

Zoals eerder vermeld (zie: 3.3) zijn er al veel voorbeelden van Nederlandse docenten die computing integreren in hun eigen vak.
Zoals: Ihsane Beyd, docente Frans, die met haar leerlingen lesjes Frans programmeert.[35]

3.7 Stel een computing-coördinator aan

Uit de Britse praktijkverhalen blijkt dat een enthousiaste en vakkundige computing-coördinator, mits goed ondersteund door bestuur en directie, essentieel is om het nieuwe onderwijs van de grond te tillen.

De ondersteuning zou dan gerealiseerd moeten worden met faciliteiten en aandacht, en mogelijk een hogere salarisschaal.

Voor meer informatie, zie: 2.5 – *Computing-coördinatoren*.

Vertaalslag

Voor Nederlandse computing-coördinatoren valt te denken aan plaatsing in een LB-functie (in het PO), of een LC- en LD-functie (in het VO).

Een voorbeeld van een Nederlandse coördinator (in het basisonderwijs) is Sandra Legters:

coördinator programmeertalen en techniek bij Stichting Openbaar Primair Onderwijs Noord Oost Achterhoek (Oponoa). Haar verhaal is opgetekend in: *Mediawijsheid op de basisschool – Succesverhalen van 21 leerkrachten* (Kennisnet & Mijn Kind Online, 2013, pp 32-34)[34].

3.8 Zoek passende beoordelings-instrumenten

'Meten' kan op twee niveaus plaatsvinden: op het niveau van het onderwijs zelf (waarbij getoetst wordt of het curriculum en de uitvoering daarvan voldoen aan de verwachtingen) en op het niveau van individuele leerlingen, om hun vorderingen te beoordelen. De vorderingen van de leerlingen kunnen ook gebruikt worden voor de beoordeling van het onderwijs zelf.

Waar het hier om gaat, is het evalueren van het onderwijs. Omdat in Groot Brittannië het computing-onderwijs nog in de kinderschoenen staat, met nieuwe lesstof en nieuwe lesmethodes, pleiten de Britten ervoor om gedegen te toetsen. Om vervolgens het onderwijs weer bij te kunnen stellen.

Een uitgebreid overzicht van beschikbare assessment-tools (beoordelings-instrumenten) en de manier waarop je ze kunt toepassen, gaat het bestek van dit rapport te buiten. Bovendien is men ook in Groot-Brittannië nog zoekende naar de juiste tools. "We focussen nu vooral op de beoordeling," zegt Chris Legg van *Eggar's School* (sec. onderwijs) in Alton. "We hebben namelijk veel leerlingen die op verschillende niveaus het nieuwe curriculum doorlopen. Omdat we dit soort onderwijs nog niet eerder hebben gegeven, moeten we goed bijhouden wat de voortgang van de leerlingen is."

Vertaalslag

We willen ons beperken tot één (nieuw) beoordelings-instrument dat veel perspectief lijkt te bieden. Volgens de Canadese *Educational Research Association*[32] kun je de leerlingen het beste een portfolio laten maken van hun werk, en dat gebruiken bij je beoordeling.

Zo'n portfolio kan een gedetailleerd beeld geven van de ontwikkeling van een leerling. Vraag de leerlingen naar hun ervaringen en de kennis die ze daarbij hebben opgedaan, en laat dat in een blog of een andere digitale vorm presenteren. Ontwikkel ontwerpscenario's die steeds verder toenemen in complexiteit. Stel telkens vast wat de sterkte van deze leerling is, en wat zijn beperkingen zijn. Maak op basis hiervan een eigen toetsingsprogramma, zodat je de vorderingen van de leerlingen kunt bijhouden en kunt afmeten aan je eigen doelstellingen en die van de leermethode.

Voor meer informatie, zie o.a.: *Werken met portfolio's* op de website 'Onderwijs Maak je Samen'.[33]

3.9 Gebruik het computing-onderwijs om je positie te versterken

Het opzetten van computing-onderwijs in Groot-Brittannië kostte veel energie en aandacht, maar het leverde ook veel op, zo blijkt. Leerlingen zijn enthousiast over het nieuwe vak, en kiezen ook steeds vaker voor een computing-specialisatie in de bovenbouw van het secundair onderwijs. Britse scholen die vroeg zijn begonnen met het aanbieden van een compleet computing-curriculum, bleken daardoor al snel populair. "Maar pas wel op dat het – vooral in het secundair onderwijs – geen jongensvak wordt," waarschuwde een van de respondenten. "Het grootste probleem dat we tot nu toe zagen, was dat informatica al snel als een typisch jongensonderwerp wordt gezien. Het aantal meisjes dat informatica wil gaan studeren valt nog steeds tegen. Maar we hopen dat we dat kunnen veranderen met computing in het primair onderwijs."

Vertaalslag

Lees ter inspiratie: *Ik kies informatica – 9 ict'sterren in beeld* (VHTO, Landelijk expertisebureau meisjes/vrouwen en bèta/techniek, 2012). Deze brochure bevat negen portretten van (vrouwelijke) leerlingen, studenten en professionals.[30]

En bedenk dat Nederland niet per se een moeilijk land is omdat er zoveel vrouwen in het onderwijs werkzaam zijn (waarvan al snel gedacht wordt dat ze weinig affiniteit hebben met computational thinking). Ten eerste moeten we blijven benadrukken dat het – met name in het basisonderwijs – vaak om heel andere dingen gaat dan programmeren (zoals: werken met speelgoed-robotjes). Ten tweede zijn er al veel vrouwelijke leerkrachten die computing integreren in hun lessen.[31]

3.10 Zorg voor een goede infrastructuur

Hoewel veel lessen zonder computer of beeldscherm (*unplugged*) gegeven kunnen worden, vooral in het primair onderwijs, staat of valt computing-onderwijs met de juiste ict-infrastructuur. Denk aan: hardware, software en internet. Besteed daarbij ook aandacht aan online-veiligheid.

Voor meer informatie, zie: 2.8 – *Infrastructuur*.

Vertaalslag

Voor het Nederlandse primair en voortgezet

onderwijs wordt hier al hard aan gewerkt, in het kader van het zogenaamde *Doorbraakproject* van de PO-Raad en de VO-raad.[36]

3.11 Zorg voor goede (bij)scholing

Het grootste probleem bij de Britse computing-docenten bleek hun gebrek aan zelfvertrouwen. De belangrijkste oorzaak: te weinig kennis en ervaring, vooral op het gebied van programmeren. De school moet dus zorgen voor: heldere leerdoelen, gedegen lesmateriaal, betrouwbare ondersteuning, en vooral: scholing.

Bijscholing in het primair onderwijs hoeft niet veel tijd te kosten, zo bleek. Denk aan een paar dagen. Bijscholing in het secundair onderwijs kan wél veel tijd kosten. Een taal als Java leer je niet in twee avonden.

In Groot-Brittannië gaat het overigens niet zo goed met de bijscholing. Dat wil zeggen: sommige scholen maken er echt werk van, maar andere niet of nauwelijks.[37] Terwijl er wel voldoende geld beschikbaar is (net als In Nederland) en er ook voldoende (bij)scholingsmogelijkheden zijn (wat in Nederland misschien minder is).

Voor meer informatie, zie: 2.6 – *Computing-docenten*.

Vertaalslag

Gedegen (bij)scholing is hoe dan ook essentieel. Ook in het SLO-rapport *Informatica in de bovenbouw havo/vwo* (2014) wordt hierop gewezen: "Intensieve bijscholing van docenten is noodzakelijk voor een kwalitatief goede uitvoering van het examenprogramma. Die bijscholing zou plaats kunnen vinden in regionale netwerken, waarin hbo, wo en bedrijfsleven een rol hebben, bijvoorbeeld via de regionale vaksteunpunten."[38]

Sluit je aan – als school of als docent – bij een of meer gebruikersgroepen. Bijvoorbeeld die van Scratch (of een van zijn dialecten voor jongere kinderen, zoals *Snap of Scratch junior*). Deel elkaars ervaringen en producten (scripts). Dat kan handig en inspirerend zijn.

3.12 Doe mee aan buitenschoolse activiteiten

Het kan enorm stimulerend werken door mee te doen (als school of als klas) aan buitenschoolse activiteiten, zo blijkt.

Volg regionale en landelijke programmeer-competities, en draag het belang van deze activiteiten zoveel mogelijk uit. In Groot-Brittannië schrijven scholen ook zelf wedstrijden uit, om het computing-onderwijs te ondersteunen.

Richt als dat kan een code- of computerclub op, waarin het computing-onderwijs wordt ondersteund in de naschoolse uren. Houd er wel rekening mee dat je op deze codeclubs en met losse code-uren geen doorlopende leerlijn kunt neerzetten, en veel minder zicht hebt op de leeropbrengsten dan met gestructureerd computing-onderwijs.

Maak de code- of computerclubs zo laagdrempelig mogelijk, adviseren de Britten. Zorg dat ze een open karakter houden, en dat ze niet alleen openstaan voor nerds en code-kunstenaars, maar voor iedereen.

Vertaalslag

Denk bijvoorbeeld aan Codeweek, Digital Champions, Safer Internet Day en Codestarter.nl, een initiatief van Nemo en Platform Betatechniek mogelijk gemaakt door Google. (zie ook: Bijlage 4.5 – *Nederlandse initiatieven*).

3.13 Profiteer van de vernieuwing

Veel docenten zijn aanvankelijk huiverig, zo blijkt. Aan de andere kant kan het ook een stimulans zijn voor docenten die graag iets nieuws willen, en hun werkgebied willen uitbreiden.
De boodschap voor schoolleiders: heb oog voor docenten die toe zijn aan iets nieuws, en geef ze de ruimte om een plan te maken (en uit te voeren).

De boodschap voor docenten: probeer je open op te stellen, en te profiteren van de kansen die er zijn. In een van de vragenlijsten die wij ingevuld retour kregen, stond bijvoorbeeld: "Toen de minister aankondigde dat er een nieuw curriculum zou komen, ben ik meteen naar het hoofd van de school gestapt, en heb ik gevraagd of ik het nieuwe curriculum voor onze school mocht bedenken en implementeren. Inmiddels ben ik Master Teacher."

Vertaalslag

In Nederland krijgen sommige docenten de ruimte om *Maker Education* oftewel 'maakonderwijs' op te zetten. Maakonderwijs is weliswaar iets anders dan computing-onderwijs, maar kan vanwege de manier waarop 'technologie' onderwezen wordt wel heel goed dienen als voorbeeld of inspiratiebron.

De Nederlandse Waag Society (Institute for art, science and technology), definieert maakonderwijs als volgt:

"Maken heeft een nieuwe betekenis gekregen. Door internet zijn nieuwe maakprincipes ontstaan, gebaseerd op openheid, sociale

verbondenheid en transparantie. Beroepen ontstaan op het grensvlak van virtuele en fysieke realiteit: de ambachten van de nieuwe eeuw. Met digitale fabricage worden fysieke producten gemaakt: de volgende fase in de digitale revolutie. Mensen willen zelf betekenis aan producten geven (personalisatie). Wanneer mensen vorm geven aan hun omgeving, neemt bewustzijn van het handelingsperspectief toe en daarmee eigenaarschap en verantwoordelijkheid voor omgeving en leven."[39]

Representanten zijn onder andere: Arjan van der Meij (www.makered.nl) en Per-Ivar Kloen (www.plakkenenknippen.nl).

3.14 Zorg voor uitwisseling, en werk samen op alle niveaus

Het Britse computing-onderwijs heeft veel baat bij het delen van kennis, ervaringen en materiaal, zo heeft CAS vanaf het begin laten zien. Iedereen kan van iedereen leren: besturen, schoolleiders, docenten, leerlingen, organisaties en bedrijven. Zorg dus dat er structureel wordt samengewerkt.

Samenwerking kan plaatsvinden op verschillende niveaus. Op beleidsniveau werd er in Groot-Brittannië bijvoorbeeld al vanaf het allereerste begin samengewerkt tussen de academische wereld (Royal Society/Academy of Sciences), het bedrijfsleven (NextGen/Microsoft) en Computing at School (CAS).

Op uitvoerend niveau werken Britse scholen samen in de CAS-hubs (zie 2.6 – *Computing-docenten*). Scholen verzinnen samen oplossingen, en docenten wisselen materiaal en ervaringen uit. Zo is er een leergemeenschap ontstaan, die als een vliegwiel functioneert voor het Britse computing-onderwijs.

Vertaalslag

Ook Nederlandse computing-docenten zijn gebaat bij samenwerking, vindt de SLO (in: *Informatica in de bovenbouw havo/vwo*, 2014), omdat ze vaak zo eenzaam zijn. Citaat: "De meeste scholen waar informatica aangeboden wordt, hebben één informaticadocent. Diens positie op school is dus vaak solitair en bijna altijd een deeltijdfunctie."[40] Voor het VO-niveau is er 'I & I', de vakvereniging voor docenten informatica & IT-coördinatoren. Die zou de handen meer ineen kunnen slaan met vertegenwoordigers van het bedrijfsleven in CodePact.

3.15 Drie aandachtspunten tot slot

Tot slot drie aandachtspunten, die verder geen vertaalslag behoeven:

Timing – In het kader van het Onderwijs2032-platform staat er in 2016 al het nodige te gebeuren. Scholen die eraan toe zijn, hebben nu een mooi moment om in te stappen. Lees de SLO-rapporten en zoek samenwerking met andere scholen.

Autonomie – In Nederland zijn scholen autonomer dan in Groot-Brittannië. Van een verplicht curriculum is hier geen sprake, al zijn er natuurlijk wel verplichte eindtermen en kerndoelen. Van die relatieve autonomie kun je gebruik maken.

Fasering – Scholen die geïnteresseerd zijn in computing, maar zich niet meteen volledig willen committeren, kunnen voorzichtig beginnen. Eerst wat lessen, en later, op basis van de ervaringen, wat meer.

4. Bijlagen

4.1 Termen en begrippen

De onderstaande definities zijn grotendeels overgenomen van de SLO.[41]

21e-eeuwse vaardigheden zijn vaardigheden – met de daarbij behorende kennis, inzichten en houdingen – die nodig zijn om te kunnen functioneren in (en bij te dragen aan) de kennissamenleving van de 21e eeuw. Centraal hierbij staan vaardigheden als zelfregulatie, creativiteit, probleem-oplossen, en samenwerken.

Een **algoritme** is een stapsgewijze oplossingsmethode (recept). Bijvoorbeeld: een afbreekalgoritme om woorden op de juiste wijze af te breken, een sorteeralgoritme om woorden of getallen in de juiste volgorde te zetten, of een compressie-algoritme om gegevensbestanden (zoals tekstbestanden, geluidsbestanden, afbeeldingsbestanden, videobestanden, etc.) te comprimeren.

Coderen is het vertalen van een algoritme naar een computerprogramma (in een specifieke programmeertaal). Coderen is een onderdeel van programmeren.

Computational thinking (computationeel denken) kan op verschillende manieren gedefinieerd worden. Formeel: "het denkproces waarmee problemen en hun oplossingen zo worden geformuleerd dat ze kunnen worden gepresenteerd in een vorm die effectief kan worden uitgevoerd door een informatie verwerkend tussenpersoon, zoals een computer of een mens, of een combinatie van beide." Dat is de oerdefinitie van Jeannette Wing[42]. In de praktijk komt het erop neer dat computational thinking een houding (manier van denken) is, waarmee je een probleem en de mogelijke oplossing in computertermen leert formuleren.

Computing is het schoolvak zoals dat o.a. in Groot-Brittannië en Ierland op de primaire en secundaire scholen gegeven wordt sinds september 2014. Aan leerlingen worden de basisprincipes van informatica, informatietechnologie en digitale geletterdheid bijgebracht, wordt uit de doeken gedaan hoe digitale systemen werken, en wordt bijgebracht hoe je deze kennis creatief kunt gebruiken door middel van programmeren.

Digitale geletterdheid (*digital literacy*) is een van de 21e eeuwse vaardigheden. Het is het vermogen om informatie te begrijpen en doelgericht te gebruiken en bestaat uit computationeel denken, basis-ict-vaardigheden, informatievaardigheden en mediawijsheid,

en het gedrag en de rol van het individu bij de digitalisering van informatie en communicatie.

Informatica (*computer science*) is de naam van de wetenschappelijke studie zoals die wordt onderwezen aan universiteiten en (technische) hogescholen. Inzichten en grondbeginselen van deze discipline staan centraal in het computing-onderwijs.

Informatica-onderwijs wordt het schoolvak (vooral bovenbouw) in Nederland meestal genoemd. Het verschil met *computing* (zie boven) is dat programmeren hier niet centraal staat, wel het algemene gebruik en de werking van computers en software.

Informatietechnologie: vak- en werkgebied waarin computers en informatica-inzichten worden toegepast in het bedrijfsleven, bij de overheid en elders.

Mediawijsheid (*media literacy*) bestaat uit de kennis, de vaardigheden en de mentaliteit waarmee burgers zich bewust, kritisch en actief kunnen bewegen in een complexe, veranderlijke en fundamenteel gemedialiseerde wereld. 'Mediawijsheid' maakt onderdeel uit van de 21e-eeuwse vaardigheden en valt onder digitale geletterdheid.

Programmeren omvat: een plan van aanpak maken, een algoritme bedenken, het algoritme vertalen naar opdrachten in een bepaalde programmeertaal (coderen), fouten opsporen en herstellen (debuggen), het programma documenteren (commentaar toevoegen) zodat andere programmeurs het kunnen begrijpen, etc.

4.2 Vragenlijst voor Britse computing-docenten

In het kader van dit onderzoek stelden wij de onderstaande vragen aan Britse computing-docenten. (Vanzelfsprekend werden de vragen in het Engels gesteld. Voor dit rapport zijn de vragen vertaald naar het Nederlands.)

Er werden ca. 400 exemplaren verzonden, waarvan er 53 beantwoord geretourneerd werden (18 van primary schools en 35 van secondary schools).

- Op welk onderwijsniveau bent u werkzaam: primair of secundair?
- Hoe heet uw school en waar die zich?
- Is de school zelfstandig of onderdeel van een scholengroep?
- Hoeveel docenten en leerlingen telt uw school?
- Hadden jullie een algemeen actieplan voor invoering van de computing-lessen?

- Hebben jullie het plan stap voor stap uitgeschreven?
- Zo ja, hoe ver zijn jullie inmiddels?
- Wat gaat er nog in de komende jaren nog gebeuren?
- Zijn er speciale computing-docenten aangetrokken of doen de huidige docenten het ernaast?
- Hoe makkelijk of moeilijk was het om het schoolbestuur te overtuigen van het nut van het nieuwe curriculum?
- Komt het nieuwe curriculum er uit dwang, of doen ze dit omdat ze al eerder enthousiast waren over het nieuwe computing-vak?
- Hoe ziet het curriculum er precies uit?
- Zijn er specifieke onderwerpen die docenten aan de diverse groepen geven of geven ze allemaal hetzelfde?
- Hoe zijn de ervaringen tot nu toe?
- Wat was makkelijk en wat was lastig voor de computing-docenten?
- Is er weerstand tegen het nieuwe curriculum? Van wie? Waarom?
- Hoe zijn jullie computing-docenten opgeleid? Hoe lang duurde dat?
- Is er ook een soort codeclub of buitenschools computer-/programmeer-initiatief op school? Hoeveel leerlingen doen daaraan mee?
- Wat is de relatie tussen het nieuwe curriculum en de computer/code-club?

4.3 Leerlijn voor het primair onderwijs (sleutelfase 2)

Hierna volgt een voorbeeld van een leerlijn voor sleutelfase 2 van het Britse primair onderwijs.

AUTUMN 1	PROGRAMMING	SYSTEMS	Lessons are once a week in year 7 and fortnightly in year 8. Lesson topics are an outline of what areas could be covered which have been split into a 4 main areas of computing. The red tabs in the corner of each cell contain URL's to resources relevant to that lesson
URL's	COMMUNICATIONS	LOGIC & DATA	

WK	DATE	Year 7	Year 8	Year 9
1	######	Intro school network, CRL resources and	Recap on games programming using scratch.	Website design and introduction to HTML.
2	######	E -safety and Cyberbullying. Introduce 'Cyber Bullying Project' Use topic to show use of search engines.	Pupils to design their own game in Scratch. Show egs of popular games. Write up Idea for HMK	Theme of web site to be decided could use WebPlus. Design storyboard.
3	######	Digital Literacy & Application skill building; DrawPlus Poster, Flash Animation (depending on group) on Cyberbullying and net safety	Start making game. This unit to tie in with D&T who will be making a 'point of sale' item to advertise the game.	Start html or WebPlus show how to set up a wed page with a master page for home.
4	######	Continue above: using DrawPlus to create a poster/Animation to make your peers aware of the Perils of Cyberbullying.		
5	######	Continue above	continue with above	Design & Create several more web pages for your site.
6	######	Digital Literacy & Application skill building; Excel to analyse questionaire on Cyberbullying and net safety campaign		
7	######	Digital Literacy & Application skill building; Excel to analyse questionaire on Cyberbullying and net safety campaign	finish off	Create & review your website.
8	######	Digital Literacy & Application skill building; Powerpoint presentation to report findings on Cyberbullying and net safety campaign		
		Monday 20th October 2013	- AUTUMN HALF TERM -	Friday 1st November 2013

AUTUMN 2		PROGRAMMING	SYSTEMS	Lessons are once a week in year 7 and fortnightly in year 8. Lesson topics are an outline of what areas could be covered which have been split into a 5 main areas of computing. The red tabs in the corner of each cell contain URL's to resources relevent to that lesson
URL's		COMMUNICATIONS	LOGIC & DATA	
WK	DATE	Year 7	Year 8	Year 9
1	######	Data representation: Bits & Bytes	Introduce: Denary to binary conversion - do worked examples _ higher students to look at base 8 & 16 resources on CRL	Algorithms: Sorting ..... see Scratch program for bubble sort
2	######	Introduce flowcharts to look at sequences to control sytems. Define flowchart symbols and their meaning.		
3	######	Use Flowol to demonstrate use of flow charts to control mimics. Use 'Flowol activites workbook' CRL: COMPUTING\ks3 resources\yr7 Logic & Data	Binary conversion use BYOB to create a system for converting binary to denary. Higher students convert to base 8 & 16	Algorithms - write in pseudo code for making a cup of tea, fixing a bike tyre
4	######	Use Flowol to demonstrate use of flow charts to control mimics;		
5	######	see Flowol activities: Lighthouse	continue with above create BYOB program to convert denary to binary numbers.	Introduce 'Progranimate' flow charting software designing and documenting programs. see http://www.progranimate.com/ This is a useful utility showing how to plan & design programs with flowcharts and pseudocode.
6	######	see Flowol activities:Level crossing		
7	######	see Flowol activities: Traffic lights	Data representation: sound and image files	as above
Monday 23rd December 2013 - CHRISTMAS HOLIDAY - Friday 3rd January 2014				

SPRING 1	PROGRAMMING	SYSTEMS	Lessons are once a week in year 7 and fortnightly in year 8. Lesson topics are an outline of what areas could be covered which have been split into a 5 main areas of computing. The red tabs in the corner of each cell contain URL's to resources relevent to that lesson
URL's	COMMUNICATIONS	LOGIC & DATA	

WK	DATE	Year 7	Year 8	Year 9
1	######	Introduction to Programming: Overview of Scratch interface (especially the colour coding system used) and how to use block to write programs.	Introduce social media - discuss the pros & cons. Twitter, Facebook & LinkedIn	Python programming intoduction to 'Idle' and the script editor. Python Basics: Reserve words(elif, def, for etc) First program, working with numbers,data types (float integers, strings, boolean) see Pyton videos: http://www.codingclub.co.uk/videos.php
2	######	Build 1st game from the eg's in in resources. Use game to expain concepts of; sequence, variable, loops and procedures. Use Scratch diary for assessment of this topic		
3	######	Continue with game programming	Look at facebook & twitter and Design a blogg	Python Basics: For and While Loops
4	######	Finish off game or look at another eg to ensure all concepts of; sequence, variable, loops and procedures are covered. Use Scratch diary for assessment of this topic		
5	######	Design you own game making use of all concepts in programming diary. This could be done using BYOB as an extension of what has been introduced in Scratch.	Task: Build a blogg	Python Basics: Conditionals if and else, Functions
6	######	Finish of game and complete Diary		
		Monday 17th February 2014 - SPRING HALF TERM - Friday 21st February 2014		

SPRING 2		PROGRAMMING	SYSTEMS	Lessons are once a week in year 7 and fortnightly in year 8. Lesson topics are an outline of what areas could be covered which have been split into a 5 main areas of computing. The red tabs in the corner of each cell contain URL's to resources relevent to that lesson
URL's		COMMUNICATIONS	LOGIC & DATA	
WK	DATE	Year 7	Year 8	Year 9
1	######	Draw a basic System and discuss components of a computer system. CPU and memory. General overview of how a computer system works	Choose an ideal computer' is a project were pupils are asked to describe all the components for a top range gaming PC. They can use the internet to complete a .ppt in CLR Computing yr9 systems. You must validate your choices. Your Budget is £3000	Networks Define: STAR/BUS/Totally Connected/LAN/WAN/MAN NETWORKS see CRL: Networks lesson
2	######	Hardware devices; look at and use of printers, scanners, tablets, control devices to appreciate I/O of a computer system		
3	######	Hardware devices; look at storage devices	...continue with project above.	How data is transferred over the Internet: Intro to networks: Warriors of the net Video See CRL: Networks lesson
4	######	Show a motherboard and get the students to complete MB handout sheet		
5	######	How to build a Rasberry Pi: Flashing SD cards and Installing Linux. SEE: ppt lesson on CRL	Intro to web page structure - style sheets, CSS, Hyperlinks	Moore's Law http://computer.howstuffworks.com/moores-law.htm Computers do not multitask - discuss see CRL: Research worksheet
6	######	Downloading Open Office and its use also see comment on; How to make your own controller		
Monday 7th April 2014 -EASTER HOLIDAY- Monday 21st April 2014				

4.4 Computing-curriculum voor het secundair onderwijs

Hieronder volgende de onderwerpen die op *Knutsford Academy in Cheshire* behandeld worden. Dit curriculum heeft betrekking op het secundair onderwijs, sleutelfases 3 en 4.

Sleutelfase 3 (11-12 jaar)

- online-veiligheid
- programmeren met Scratch en Small Basic
- besturing (stroomdiagrammen, verkeerslichten, Flowol control software)
- webpagina-ontwerp met HTML

Sleutelfase 4 (13 jaar en ouder)

- programmeren met Python
- programmeren met PHP
- database-ontwerp met MySQL
- logische gates en waarheidstabellen
- binaire getallen, binair optellen en aftrekken

Aanvullende onderwerpen voor leerlingen die *GCSE Computer Science* kiezen:

- de *software development lifecycle*
- hexadecimale getallen
- de binaire representatie van geluid
- de binaire representatie van beeld
- netwerken en internet (protocollen, standaarden, IP-adressen)

4.5 Nederlandse initiatieven

BIT-kids: richt zich op programmeren voor kinderen vanaf 10 jaar. “Wij geloven dat iedereen kan programmeren, dat het leuk is en ook nog een extreem nuttig.” BIT-kids geeft inleidende workshops en kan in overleg een serie programma’s opstellen op lokatie of op school. Zie verder: bIT-kids.nl

Bomberbot: een soort ‘Scratch’ voor Nederland. Met als motto: “Breng de leerlingen programmeervaardigheden bij voor het leven”. Zie verder: www.bomberbot.com

Code Cult (Amsterdam)**:** in tegenstelling tot traditionele vormen van informatica-onderwijs, staat bij Code Cult de leerling centraal, die zelf kiest welk pad zij of hij wil volgen. Code Cult ontwikkelt lessen die erop gericht zijn leerlingen zelf dingen te laten ontdekken en vindt het belangrijk dat ze vaardigheden ontwikkelen op een ongedwongen en vrije manier. Zie verder: codecult.nl

Code Studio: internationale website voor leerlingen en docenten. Bevat cursussen, een speellaboratorium, en mogelijkheden om spellen te maken met Flappy Code. Veel materiaal is in het Nederlands vertaald. Zie verder: studio.code.org

Codekinderen: divers lesmateriaal voor kinderen op de basisschool. “Codekinderen laat basisschoolleerlingen van groep 3 tot en met 8 kennis maken met de ‘achterkant’ van de apparaten die ze dagelijks gebruiken.” Ook lesmateriaal voor leerkrachten.
Zie verder: www.codekinderen.nl

CodePact: onder het motto ‘Alle kinderen digitaal vaardig’ wil CodePact – onderdeel van TechniekPact – kinderen de kans geven om te leren programmeren en digitaal geletterd te maken, o.a. door voorbeeldleerlijnen te maken, scholen te voorzien van goede technische infrastructuur, en docenten te professionaliseren. Dit doet zij door publieke en private partijen bij elkaar te brengen, zoals het Ministerie van OC&W, Microsoft, Malmberg, IBM, Google en Randstad. Het programma loopt van 1 april 2015 tot 1 januari 2017. Zie verder: codepact.org

CoderDojo (Amsterdam)**:** een open source, door vrijwilligers opgezette beweging die staat voor het houden van gratis, non-profit development clubs en reguliere sessies voor jonge mensen. Omdat CoderDojo open source is kan elke Dojo anders zijn en staat volledig los van andere Dojo’s. Bij een Dojo leren jonge mensen, tussen de 5 en 17 jaar, om te programmeren, websites bouwen, apps ontwikkelen, programma’s te maken, games te maken en meer. Zie verder: www.coderdojo.nl

CoderDojo for Girls (Amsterdam)**:** Digivita voert activiteiten uit om meisjes te enthousiasmeren en informeren over ict-opleidingen en beroepen. digiVita is een project van VHTO, Landelijk expertisebureau meisjes/vrouwen en bèta/techniek. Zou je ook zelf wel willen leren programmeren en bijvoorbeeld apps ontwikkelen? Houd dan de site van Digivita in de gaten, want digiVita gaat binnenkort workshops ‘coding’ (=programmeren) organiseren, speciaal voor meisjes! Op een paar plekken in het landen vinden ook al dit soort workshops plaats.
Zie verder: digivita.nlcoderdojo-girls

CoderDojo Rotterdam: verzorgt gratis lessen / workshops om kinderen van 7 - 18 jaar te leren programmeren. Er worden regelmatig evenementen georganiseerd waar kinderen en hun ouders kunnen samenkomen om te werken aan websites, games, apps en software. Het hele project wordt door vrijwilligers opgezet!
Zie verder: www.coderdojo-rotterdam.nl

Codestarter.nl: Codestarter is een initiatief van Science Center NEMO en Platform Bèta Techniek en wordt mogelijk gemaakt door Google.

Het brengt het aanbod van naschoolse activiteiten in kaart voor kinderen in de leeftijd van 8 tot 12 jaar op het gebied van programmeren, coderen en 3D-printen. Zie verder: codestarter.nl

CodeUur: de Stichting CodeUur wil kinderen van groep 7 en 8 van de basisschool digitaal vaardig maken. “Wij brengen programmeren echt in de klas.” Stichting die zich ervoor beijvert om basisscholen “digitaal vaardig te maken”. Dit willen ze doen door in groep 7 en 8 van de basisschool het hele jaar door “programmeren structureel de klas in te brengen”. Het motto van deze stichting: Elke vrijdag één uur programmeren. Op de website staat lesmateriaal en kunnen leerlingen kijken waar ze les kunnen krijgen. Ook degene die les willen geven in programmeren kunnen zich hier aanmelden. Zie verder: www.codeuur.nl

Codeweek: is een landelijke campagne die leerlingen en docenten in het PO en VO laat kennismaken met programmeren. Ook moet de Codeweek kinderen stimuleren zelf aan de slag te gaan met programmeren. In de Codeweek worden door het land diverse activiteiten rondom het thema ‘code’ georganiseerd. Zie verder: codeweek.nl

De Bonte Bouwplaats (Delft)**:** organiseert kinderfeestjes rondom het thema ‘bouwen’. Denk bijvoorbeeld aan legofeestjes, maar ook aan Mindstorm feestjes waarbij je een robot leert besturen via door jou zelfgeschreven programma’s. Zie verder: debontebouwplaats.nl

Digi-lab: verzorgt workshops, clinics, lessen, cursussen en projectondersteuning voor leerlingen, studenten en volwassenen op lokatie. Wat kun je maken in het digi-lab? Animaties 2d en 3d, filmpjes, digitale prentenboeken, stripverhalen, slideshows, fotocollages, hoorspelen, zelfgeschreven multimediale musicals, je eigen games en interactieve verhalen in Scratch (ook met Lego WeDo, picoboard en MaKey). Zie verder: www.digi-lab.org

DigiVita: VHTO organiseert ‘DigiVita Code Events’ in verschillende steden in Nederland om meisjes te leren programmeren. Tijdens de DigiVita Code Events leren meisjes van 8 tot 18 jaar programmeren van vrouwelijke ict-studenten en ict-professionals. http://www.vhto.nl/projecten/digivita/digivita-code-events/

Digital Playground: geven workshops (ook op scholen) waarbij het maken van eigen nieuwe media centraal staat. Zie verder: www.digitalplayground.nl

Doorbraakonderwijs en ict:
het *Doorbraakproject* is een gezamenlijk initiatief van de PO-Raad, VO-Raad en de ministeries van OC&W en Economische Zaken. Vragers (schoolbesturen) en aanbieders (educatieve uitgevers, software-leveranciers, en distributeurs) worden hierin bijeengebracht. Er wordt vooral gezocht naar ict-maatwerk. Zie verder: doorbraakonderwijsenict.nl

FabLab Benelux: op deze website is informatie en links te vinden naar alle FabLabs in Nederland. FabLabs zijn publieke toegankelijke plaatsen waar je kunt werken met digitale fabricage tools zoals 3d printers en lasercutters, maar vaak ook met micro-computers en andere digitale middelen. FabLabs zijn verspreid over heel Nederland. Veel FabLabs hebben ook onderwijsprogramma's. Zie verder: fablab.nl

FabSchool (Amsterdam)**:** speerpunten binnen de FabSchool workshops zijn creativiteit, techniek en ondernemerschap. Door deel te nemen aan de sessies ontwikkelen jongeren onder andere hun probleemoplossend vermogen. Ze ontdekken dat ze ogenschijnlijk grote problemen zelf kunnen oplossen, met behulp van technologie. Er wordt uitgelegd hoe technologie in elkaar zit en welke processen er achter technologische oplossingen schuilgaan. De workshops sluiten altijd aan bij de leefwereld van de deelnemers. Zie verder: waag.org/nlproject/fabschool

First Lego League: een wedstrijd die jongeren tussen de 9 en 15 jaar uitdaagt om de maatschappelijke rol van techniek en technologie te onderzoeken aan de hand van verschillende opdrachten. Zie verder: firstlegoleague.nl

FryskLab (Friesland)**:** een initiatief van Bibliotheekservice Fryslân. FryskLab is ondergebracht in een voormalige bibliobus en het eerste officiële bibliotheek-FabLab in Europa. Met FryskLab onderzoekt BSF hoe de inzet van een mobiel FabLab binnen de onderwijssituatie bijdraagt aan de creatieve, technische en ondernemende vaardigheden van kinderen en jongeren. Het lesaanbod laat scholieren kennismaken met digitale fabricage gekoppeld aan 21st Century Skills. Zie verder: www.frysklab.nl

GameWizards: via cursussen en/of workshops kinderen kennis te laten maken met de wereld van software ontwikkeling. Binnen GameWizards werken professionals op het gebied van techniek en creativiteit samen. Zo ontstaat de perfecte mix om kinderen te leren hoe ze zelfstandig games kunnen ontwikkelen. Zie verder: gamewizards.nl

Junior Techniek: geeft workshops techniek en programmeren. Scratch is een prima invulling voor de les techniek. De kinderen leren niet alleen programmeren maar ook samenwerken. Dit doen ze door op een eenvoudige en slimme manier games, tekenfilmpjes en meer te maken en hierna hun werk te delen met anderen. Hierdoor leren ze van andere Scratchers van over de hele wereld. Zie verder: www.junior-techniek.nl

Kodekup: laat kinderen tijdens de Europese Codeweek hun eigen game maken met de Kode Game Lab Tools. Zie verder: www.kodukup-europe.org

Lego Leerlijn: ondersteunt de lessen en activiteiten met robotica door middel van gratis werkbladen voor de LEGO® Education WeDo en de LEGO® MINDSTORMS® Education EV3 robots waarmee leerlingen zelfstandig aan de slag kunnen. Zie verder: www.legoleerlijn.nl

Mediawijsheid.nl: deze 'wegwijzer in Medialand' is een initiatief van Mediawijzer.net (zie aldaar) en bedoeld voor ouders, jongeren, (semi-)overheid, pers, onderwijs, en iedereen die meer wil weten over mediawijsheid. Informeert o.a. over cyperpesten, gaming, grooming, online privacy, sociale media en de digitale kloof. Zie verder: www.mediawijsheid.nl

Mediawijzer.net: netwerk-organisatie, opgezet op initiatief van de ministeries OC&W en het toenmalige Jeugd en Gezin, om Nederlandse burgers en organisaties mediawijs te maken. Zie verder: www.mediawijzer.net

MOOC MEE!: verzorgt 'MOOCs' (Massive Open Online Courses) voor leerkrachten, mediacoaches en ouders die meer willen weten over programmeren. Zie verder: www.moocmee.nl

Ons Onderwijs2032: het Platform Onderwijs2032 zoekt naar antwoorden op vragen als: Wat moeten kinderen nú leren om straks een vliegende start te kunnen maken op de arbeidsmarkt? Welke kennis en vaardigheden hebben zij nodig om optimaal te kunnen functioneren in de samenleving van de toekomst?Zie verder: onsonderwijs2032.nl

RISE OF //CODE (Amsterdam)**:** een initiatief dat Nederlandse leerlingen kennis laat maken met programmeren, hardware, 3D printen en meer! Alle docenten zijn professionals uit de Amsterdamse startup scene. Zie verder: riseofcode.com

Robomind: educatieve programmeeromgeving voor kinderen in het primair én het secundair onderwijs. Zie verder: www.robomindacademy.com

ROFFAB (Rotterdam)**:** realiseert een netwerk van werkplaatsen verspreid door Rotterdam waar kinderen zelf kunnen experimenteren met nieuwe technologieën. Denk hierbij aan 3D-printen en robotica. Technologie die de wereld om je heen slimmer en leuker maakt. Zie verder: www.roffab.nl

ScratchMatch: wedstrijd voor het basisonderwijs en voortgezet onderwijs om het gaafste Scratch-project te maken. Zie verder: www.scratchmatch.nl

Scratchweb: website (in het Nederlands) rond het populaire en op vele scholen gebruikt MIT-programma Scratch. Er is een aparte sectie aan Scratch-onderwijs gewijd, met lesmateriaal, vertalingen en nieuws. Verder zijn er voorbeelden, projecten, gebruikersverhalen en is er nieuws over boeken, software, codeclubs en websites. Zie verder: www.scratchweb.nl

SETUP (Utrecht)**:** organiseert bijeenkomsten (presentaties, workshops, tentoonstellingen) over technologie. Voor kinderen en jongeren, door jongeren. Zie verder: www.setup.nl

Versnellingsvragen PO-Raad: vanuit het project 'Beter en slimmer leren met ict' helpt de PO-Raad schoolbesturen hun ambities versneld waar te maken en belemmeringen weg te nemen. Zie verder: www.poraad.nl/themas/innovatie-ict/onderwijs-en-ict/wegnemen-belemmeringen-0

Waag Society – Fabschool Kids (Amsterdam)**:** leert kinderen programmeren met 'Scratch'. Zie verder: waag.org/nl/event/fabschool-kids-programmeren-met-scratch

4.6 Lessen voor het basisonderwijs (voorbeeld: Codekinderen)

Het gratis lespakket *Codekinderen* (www.codekinderen.nl) van Kennisnet bevat aantrekkelijke lessen voor het basisonderwijs. Alles komt aan bod: van digitaal spelen tot echt programmeren. Dit lespakket is een goed voorbeeld van computing-onderwijs op de basisschool.

De lessen zijn verdeeld in drie categorieën: Unplugged (geen computer nodig), Maken, en Programmeren. Elke les bevat instructies voor de leerlingen en een docentenhandleiding. Inclusief filmpjes, werkbladen en handige links. Hieronder volgt een overzicht van de beschikbare lessen.

Unplugged

- binair tellen
- pixels tekenen

- robot-taal
- sandwich robot

Maken

- stripverhalen maken (met Bitstrips)
- 3D-tekeningen maken (met Blokify)
- 3D-bouwplaten maken (met Foldify)
- eenvoudige games maken (met GameStudio)
- eenvoudige én geavanceerdere games maken (met GameMaker)
- stop-motion animaties maken (met Lego Movie)
- je eigen toetsenbord maken (met Makey Makey)
- eenvoudige animaties maken (met Pivot Animator)
- klikbare protoypes maken (met Pop)

Programmeren

- een robotje programmeren (met Bee-Bot)
- code schrijven in JavaScript (met CodeMonster)
- filmpjes, interactieve tekeningen, quizzen en spelletjes maken (met HopScotch)
- een vogel door een doolhof sturen (met Ko de Kraker)
- een robotje over tegelplateaus manoeuvreren (met Lightbot)
- een robotje aansturen (met RoboMind)
- programmeren met de populairste programmeertaal voor kinderen (Scratch)
- een biggetje laten pakken door een boze vogel (met Hour of Code)

4.7 Lessen voor het voortgezet onderwijs

Ict-coördinator Frans Peeters (Ostrea Lyceum, Goes) onderhoudt de website informaticaVO.nl. De menukeuze 'Lesmateriaal' geeft toegang tot een veelheid aan lessen, over elk denkbaar onderwerp. Van mediawijsheid tot hardware-projecten en programmeertalen.

4.8 Naschoolse computing-lessen

Kevin McLaughlin van de *Old Mill Primary School* in Leicestershire organiseert al enige jaren een naschoolse *computing club*. In de nieuwsbrief van Computing at School (zomer 2014) legt hij uit hoe hij dit doet en waarom.[43]

"Toen ik begon met deze naschoolse *computing club*, was hij meteen al overtekend. Alleen de eerste 20 kinderen kon ik toelaten. Het zijn ook niet alleen jongens; er zijn 6 meisjes en later in het jaar komen er misschien nog een paar bij."

Mc Laughlin wilde de kinderen geen *Scratch* leren, omdat dat al in het officiële computing-curriculum gebeurt. Hij maakte gebruik van learn.code.org omdat daar een gestructureerde introductie tot informatica stond in 20 stadia. In elk stadium zijn er mini-activiteiten voor de kinderen, met daarbij kleine onderscheidingen en prijzen. Voor elk kind

is er een eigen inlog en een eigen traject dat door de coördinator en het kind kan worden gevolgd. Soms werken de kinderen samen aan een project, waarbij het telkens ging om problemen die met behulp van programmatuur moesten worden opgelost. Het kind dat het eerst een groot probleem kon oplossen kreeg een trofee.

"Als coördinator kan ik de voortgang van de klas en de individuele leerlingen volgen en online commentaar geven," zegt McLauglin. "Sommige kinderen gaan heel snel door het materiaal, anderen hebben een langzamer tempo."

4.9 Boeken over 'programmeren voor kinderen'

- Adams, R., Bell, T., Fellows, McKenzie J.,M., Witten, I.H., *Computer Science Unplugged*, CSUnplugged.com, Nieuw-Zeeland, 2010.
  Dit boek bevat tal van informatica-lessen voor kinderen, die allemaal zonder computer (unplugged) kunnen worden uitgevoerd.
  (6-8 jaar) csunplugged.org

- Beer, S. de, *Apps maken met meneer de beer*, Mainpress 2013.
  In dit boek laat Serge De Beer zien dat iedereen een app kan leren maken, zonder te hoeven programmeren. Je begint met het downloaden van het gratis en eenvoudig te installeren programma GameSalad. Alle belangrijke stappen komen aan bod tot het moment dat je je app in de App Store kunt gaan zetten. Voor docenten is een speciale handleiding beschikbaar met handige weetjes en tips voor toepassing in lessituaties.

- Bird, James, Caldwell, Helen, Mayne, Peter, *Lessons in Teaching Computing in Primary Schools*, Learning Matters, 2014
  Dit boek begeleidt het Britse computing curriculum en bevat – naast een overzicht van de lesstof – tal van voorbeelden voor computing-lessen. Geeft ook inzicht in hoe je hier les in zou kunnen geven.

- Cleijn, C., *GameMaker voor kids*, Van Duuren Media, 2011
  GameMaker is een programma waarmee je heel gemakkelijk allerlei soorten eenvoudige, maar ook uitgebreide computerspellen kunt maken. Dit boek laat kinderen zien hoe dat moet. Het eerste hoofdstuk vertelt hoe GameMaker (gratis in een lite versie) gedownload en geinstalleerd moet worden, en geeft informatie over het

werken met afbeeldingen en het vinden van gratis (rechtenvrije) plaatjes en geluiden op internet. Daarna volgt direct de praktijk. (8-14 jaar)

- Cleijn, C., *Sketchup voor kids*, Van Duuren Media, 2011
  Het principe van SketchUp is: alles is 3D. Dit praktische werkboek, voorzien van voorbeelden en illustraties, laat kinderen zien hoe je zelf driedimensionale tekeningen maakt. "De meeste tekenprogramma's die je kent, zijn tweedimensionaal. Je kunt er een tekening in maken, maar het blijft plat. SketchUp verandert je platte tekening in een 3D-tekening. Simpel gezegd: je tekent een vierkant en SketchUp maakt er een kubus van. Daarvoor heeft SketchUp speciale gereedschappen. Vanuit dit idee kun je hutten bouwen, huizen metselen, meubels timmeren, een zwembad in de tuin graven, en ga zo maar door." (8-14 jaar)

- *CodeKlas*
  Een inspiratieboek over programmeren met kinderen en jongeren, voor docenten en ouders. Rond 8 thema's (zoals robots, games, en websites) beschrijven Pauline Maas en 40 co-auteurs gereedschappen waarmee iedereen direct aan de slag kan. In 10 interviews vertellen mensen uit het onderwijs en het bedrijfsleven waarom zij het belangrijk vinden dat kinderen al op jonge leeftijd in aanraking komen met programmeren. codeklas.nl

- *Beginnershandleiding Scratch*
  In het Nederlands vertaalde handleiding van de educatieve programmeertaal Scratch (zie ook: Bijlage 4.11 – Veel gebruikte software en hardware) drive.google.com/file/d/0B_L_APuWYxGfZHN3VU91Vlp0b1U/view

- Ford, R., *Leren programmeren met Scratch*, Pearson Benelux, 2013
  In dit boek leren kinderen hoe je stap voor stap met Scratch een computerprogramma kunt maken. "Je leert hoe je de computer opdrachten geeft waarmee hij precies doet wat jij wil. Je leert hoe je poppetjes op het scherm laat bewegen, hoe ze geluid kunnen maken, en hoe ze kunnen reageren op wat jij doet." In 6 projecten laat de auteur een kat rennen, een spin dansen, een ster ontploffen, en nog veel meer. (8-12 jaar)

- Libow Martinez, Sylvia en Stager, Gary, *Invent To Learn: Making, Tinkering, and Engineering in the Classroom*, Constructing Modern Knowledge Press, 2013

Een boek dat speciaal bedoeld is voor docenten en onderwijskundigen over 'maak'-lessen en actieve lesopvattingen. Bevat tal van tips, oefeningen en inzichten voor actief onderwijs.

- McCue, Camille, *Coding For Kids For Dummies*, John Wiley & Sons, Hoboken NJ, 2014
  Deeltje uit de zeer populaire 'For Dummies'-serie, waarin uitgebreid wordt verteld hoe kinderen kunnen leren programmeren. Met tal van kleine projecten waarin telkens een onderdeel van het programmeren wordt behandeld. (6-14 jaar)

- Peters, V.G.B., *Programmeren voor kinderen met Flash*, Bruna, 2003
  Al wat ouder boek over het programmeren met het animatieprogramma Flash dat langzamerhand zelf ook verouderd aan het raken is. In het boek wordt stap voor stap uitgelegd hoe je in Flash MX je eigen animaties en spellen maakt. (8-14 jaar)

- Steinhauser, P.L., Mousetracks: *A Kid's Computer Idea Book*, Tricycle Press, 2004
  Dit boek is georganiseerd rond populaire thema's als sport, tropisch regenwoud, ruimtevaart en uitgeven. Bevat zo'n 70 stap-voor-stap-activiteiten die je met elke graphics en tekstverwerkingssoftware op elke soort computer kunt uitvoeren. (6-10 jaar)

- Vorderman, C., *Programmeren voor kinderen - Leer stap voor stap programmeren en je eigen computergames maken*, Lannoo, 2014
  Dit boek staat vol achtergrondinformatie met heldere voorbeelden en oefeningen. Een gids met kleurrijke illustraties en stap-voor-stap instructies om programma's te schrijven in Scratch en Python. (4-12 jaar)

4.10 Veel gebruikte software en hardware

Software

Baltie is een educatief programmeerhulpmiddel dat vooral in educatieve omgevingen wordt gebruikt om kinderen te leren programmeren. Dit hulpmiddel maakt gebruik van een personage, de tovenaar Baltie, die de gebruiker bijstaat en uitdaagt. www.sgp.cz/en/whatisbaltie.asp

Blender is een open source en gratis 3D-pakket dat veel op scholen in de Verenigde Staten wordt gebruikt. www.blender.org

DreamSpark is een programma waarmee leerlingen van alle leeftijden gratis Microsofts professionele ontwikkeltools kunnen gebruiken als Visual Studio Professional, XNA, Expression Suite, Windows Phone Developer Tools en SQL Server. www.dreamspark.com

Kodu is een compleet hulpmiddel om games mee te maken, gebaseerd op een unieke eigen taal die met behulp van icoontjes wordt bediend. Het pakket richt zich op leerlingen en levert een introductie in de logische en probleemoplossende activiteiten die bij software-ontwikkeling komen kijken. www.kodugamelab.com

Lego Mindstorm NXT is een basic-programmeertaal die gebruikt kan worden door leerlingen die dingen willen ontwerpen. Het programma bevat een 3D-editor waarmee je virtuele omgevingen kunt ontwerpen. www.lego.com/nl-nl/mindstorms/support

Logo is een van de oudste educatieve programmeertalen. Hij werd ontwikkeld door de MIT-onderzoeker en onderwijskundige Seymour Papert. Er zijn inmiddels tal van varianten van dit programma, waaronder XLogo (Java), Logo Codo (web) en Visual Logo. nl.wikipedia.org/wiki/Logo_(programmeertaal)

Python is een (geïnterpreteerde) programmeertaal die gebruik maakt van 'inspringen' om de structuur van het programma aan te geven, vergelijkbaar met het noteren van algoritmes in pseudocode. Python kan op een basisniveau gebruikt worden maar ook op een geavanceerde manier door gevorderde gebruikers. De taal is heel geschikt voor scholen, open source, en gratis. De downloadable pakketten kunnen zo samengesteld worden dat je ze op vele manieren in het curriculum kunt inzetten, bijvoorbeeld de 3Dgraphics-module bij natuurkunde en wiskunde. www.python.org

Scratch is een programmeertaal die speciaal werd ontwikkeld voor kinderen, door het invloedrijke Massachusetts Institute of technology (MIT). In Groot-Britannië en Nederland veel gebruikt op scholen en in code- of computerclubs. Je maakt programma's door blokken in elkaar te klikken zoals met Lego. Binnen een paar minuten is het eerste resultaat zichtbaar. scratch.mit.edu

Hardware

Arduino: een mini-PC van een paar tientjes, op een klein printplaatje (zonder behuizing), vergelijkbaar met de Raspberry Pi (zie aldaar). De Arduino is vooral geschikt voor het uitlezen van sensoren en het aansturen van apparatuur.

Heel geschikt voor scholen.
nl.wikipedia.org/wiki/Arduino_(computerplatform)

Bee-Bots, Roamers, Pro-Bots en Big Traks: educatief programmeerbaar speelgoed (robots) dat vooral door jonge kinderen gebruikt kan worden om spelenderwijs de grondbeginselen van informatica te leren kennen.
www.b-bot.nl/webshop/educatief-speelgoed

Einstein Tablet+: een leermiddel in de vorm van een Android-tablet, speciaal voor vakken als natuurkunde en biologie. De tablet beschikt over sensoren die luchtvochtigheid en uv-straling meten. Docenten kunnen via de Einstein Activity Maker hun eien experimenten maken, delen en downloaden. www.einsteinworld.com

Picaxe: microbot en software waarmee kinderen hun eigen robot kunnen leren programmeren.
www.picaxe.com

Raspberry Pi: een mini-PC van een paar tientjes, op een klein printplaatje (zonder behuizing), vergelijkbaar met de Arduino (zie aldaar). De Raspberry Pi is geschikt voor video- en algemene computer-toepassingen. Heel geschikt voor scholen. www.raspberrypi.org

4.11 Vereisten voor een computing-docent

Om te achterhalen wat de taken en verantwoordelijkheden van Britse computing-docenten zijn, kun je vacatures bekijken. Zoek bijvoorbeeld op *"Teacher of ict and Computing" "job description"*. Een standaard-voorbeeld volgt hieronder.

1. Teaching and learning

- To implement and deliver an appropriately broad, balanced, relevant and differentiated curriculum for ict and Computing
- To monitor and support the overall progress and development of students
- To facilitate and encourage a learning experience which provides students with the opportunity to achieve their individual potential
- To contribute to maximizing of student attainment
- To share and support the school's responsibility to provide and monitor opportunities for personal and academic growth

2. Planning

- To assist in the development of appropriate schemes of work towards selected ict and Computingsyllabuses. To develop teaching resources, schemes of work, marking policies and teaching strategies in the curriculum area

- To contribute to the curriculum area and department's development plan and its implementation
- To ensure department documentation and practice in relation to Health and Safety is in place and adhered to

3. Curriculum Development

- To assist in the process of curriculum development in ict and Computingso as to ensure the continued relevance to the needs of students, examining and awarding bodies and the school's ethos

4. General Duties

- Participate in relevant meetings with colleagues, parents and be involved in links with external agencies as part of curriculum enhancement, including extra-curricular activities and off-site visits
- Support other members of the curriculum area and students as appropriate
- Review your own professional development and maintain professional standards
- Carry out any other such duties as the post holder may reasonably be required to do

5. Safeguarding

- To follow the school's policy in respect of safeguarding and child protection and ensure the health and safety of the students

Bibliografie

(geen auteursvermelding), *Bestuursakkoord voor de sector primair onderwijs*, OCW/PO-Raad, juli 2014. www.poraad.nl/files/over_de_po-raad/bestuursakkoord_po.pdf

(geen auteursvermelding), *Computing programmes of study: key stages 1 and 2.* National curriculum in England, Department of Education, DFE-00171-2013, 2013. www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

(geen auteursvermelding), *Klaar voor de toekomst! Samen werken aan onderwijskwaliteit / Sectorakkoord VO 2014-2017*, VO-Raad en OCW, april 2014. www.vo-raad.nl/userfiles/bestanden/Sectorakkoord/Sectorakkoord-VO-OCW.pdf

(geen auteursvermelding), *Programmeren op de basisschool?*, Verenigde Scholen J.A. Alberdingk Thijm, 2012. www.atscholen.nl/de_organisatie/ICT_innovatie/Leren_Programmeren/Paginas/default.aspx

(geen auteursvermelding), *Mediawijsheid. De ontwikkeling van nieuw burgerschap*, Raad voor Cultuur, juli 2005. www.rijksoverheid.nl/documenten-en-publicaties/kamerstukken/2006/09/13/bijlage-c-advies-raad-van-cultuur-mediawijsheid-de-ontwikkeling-van-nieuw-burgerschap.html

(geen auteursvermelding), *Subject knowledge requirements for entry into computer science teacher training – expert group's recommendations,* Department of Education, 2012. www.computingatschool.org.uk/data/uploads/CSSubjectKnowledgeRequirements.pdf

(geen auteursvermelding), *The national curriculum in England Key stages 1 and 2 - framework document,* Department for Education, sept. 2013. www.gov.uk/government/uploads/system/uploads/attachment_data/file/335133/PRIMARY_national_curriculum_220714.pdf

Adams, R., Bell, T., Fellows, McKenzie J.,M., Witten, I.H., *Computer Science Unplugged*, CSUnplugged.com, Nieuw-Zeeland, 2010. csunplugged.org/sites/default/files/activity_pdfs_full/unpluggedTeachersMar2010-USletter.pdf

Andrews, R., et al., *School Curriculum Differences* across the UK, National Foundation for Educational Research, 2001. www.leeds.ac.uk/educol/documents/00003560.htm

Bagge, Phil, *Computational Thinking in the Primary Computing-curriculum*, Sept. 2014. philbagge.blogspot.nl/2014/09/computational-thinking-in-primary.html

Bagge, Phil, Junior *Computer Science, School Computer Network*, Code-it.co.uk, 2013. code-it.co.uk/internet/schoolsnetwork_planning.pdf

Balanskat, A., Engelhardt, K., Computing our future - *Computer programming and coding -Priorities, school curricula and initiatives across Europe,* European Schoolnet, okt 2014. www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887

Barr, V. & Stephenson, C., 'Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?' In: *ACM Inroads,* vol. 2, no. 1, 2011, pp. 48-54. csta.acm.org/Curriculum/sub/CurrFiles/BarrStephensonInroadsArticle.pdf

Barthel, P., Brock, B. de, Jong, F. de, Lagendijk, I., Smeets, D. en Korbijn, A., *Digitale geletterdheid in het voortgezet onderwijs. Vaardigheden en attitudes voor de 21ste eeuw.* Amsterdam: KNAW, 2012. www.knaw.nl/shared/resources/actueel/publicaties/pdf/20121027.pdf

Berry, M., *Computing in the national curriculum. A guide for primary teachers.* CAS/Naace, 2013. www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf

Berry, M., *QuickStart Computing; a CPD toolkit for primary teachers,* Computing at School, 2014. www.quickstartcomputing.org

Berry, M., *QuickStart Computing; a CPD toolkit for secondary teachers,* Computing at School, 2014. www.quickstartcomputing.org

Bitto, D. & Mirolo, C., 'Archaeology of Information in the Primary School'. In: I. Diethelm, R. Mittermeir & T (eds.): *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages*, Springer Berlin Heidelberg, 2013, pp. 115-126. link.springer.com/chapter/10.1007/978-3-642-36617-8_10

Boot, L., *Kinderen leren programmeren is schitterend,* Kennisnet, mei 2014
innovatie.kennisnet.nl/kinderen-leren-programmeren-is-schitterend

Bradshaw, P. & Woollard, J., 'Computing at school: an emergent community of practice for a re-emergent subject'. In: P*roceedings of the International Conference on ict in Education* (ICICTE 2012), Rhodos, 2012.
www.icICTe.org/Proceedings2012/Papers/15-2-Bradshaw.pdf

Brennan, K., & Resnick, M., *New frameworks for studying and assessing the development of computational thinking.* American Educational Research Association meeting, Vancouver, BC, Canada, 2012.
web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Brons, Th., 'Het Nederlandse ict-onderwijs is achterhaald'. In: *Emerce*, 27 november 2014
www.emerce.nl/opinie/nederlandse-ICTonderwijs-achterhaald

Brown, N., et al, 'Bringing Computer Science Back into Schools: Lessons from the UK'. In: *SIGCSE Symposium 2013*, Denver, CO, mei 2013.
research.microsoft.com/en-us/um/people/simonpj/papers/cas/sig132-brown.pdf

Brown, N., Sentance, S., Crick, T., and Humphreys, S. 2014. 'Restart: The Resurgence of Computer Science in UK Schools'. In: *ACM Transactions on Computing Education* (TOCE), Volume 14 Issue 2, New York, June 2014, Art. 9, p. 1-22.
dl.acm.org/citation.cfm?id=2602484

Computing at School, *A National Working Group*, Cambridge, aug 2010.
research.microsoft.com/en-us/um/people/simonpj/papers/cas/computingatschoolcacm.pdf

Computing at School Working Group, Computer Science: *A Curriculum for Schools, Computing at School*, 2012.
www.computingatschool.org.uk/data/uploads/Computing_at_School.pdf

CAS/Computing at School, *Running a CAS Hub - Operations Manual v4*, Computing at School, 2013.
www.computingatschool.org.uk/data/uploads/Hub_OperationsManual.pdf

Curzon, P., et al., *Developing computational thinking in the classroom: a framework*, Computing At School, 2014.
community.computingatschool.org.uk/files/3517/original.pdf

Crick, T., & Sentance. S., 'Computing at School: Stimulating Computing Education in the UK'. In: *Proceedings of the 11th Koli Calling International Conference on Computing Education Research,* 2012.
drtomcrick.com/2012/01/15/cas-at-koli-calling-2011

Department for Education (DFE), corp creators, *Subject knowledge requirements for entry into computer science teacher training : expert group's recommendations,* Teaching Agency, nov 2012.
dera.ioe.ac.uk/15780

Fisser, P., Hoeven, M. v.d., Thijs, A., *Digitale geletterdheid en 21e eeuwse vaardigheden in het funderend onderwijs: een conceptueel kader,* SLO, jan. 2014.
www.slo.nl/downloads/documenten/digitale-geletterdheid-en-21e-eeuwse-vaardigheden.pdf

Furber, Steve, Shut Down or Restart? *The Way Forward for Computing in UK Schools,* The Royal Society, London, jan. 2012.
royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Gee, James Paul. *What video games have to teach us about learning and literacy.* New York, 2003.
en.wikipedia.org/wiki/What_Video_Games_Have_to_Teach_Us_About_Learning_and_Literacy

Humphreys, S., Mitchell, B., Peyton Jones, S., *Computing at school in the UK: from guerrilla to gorilla.* Submitted to CACM, may 2013.
research.microsoft.com/en-us/um/people/simonpj/papers/cas/ComputingAtSchoolCACM.pdf

Humphreys, S., *Lead School Audit 2014 Report*, Computing At School, 2014.
community.computingatschool.org.uk/resources/3044

Kemp, P., *Computing in the national curriculum - a guide for secondary teachers,* Naace / Computing at School, 2013.
www.computingatschool.org.uk/data/uploads/cas_secondary.pdf

Libow Martinez, Sylvia, Stager, Gary, I*nvent To Learn: Making, Tinkering, and Engineering in the Classroom,* Constructing Modern Knowledge Press, 2013.
www.inventtolearn.com

Livingstone, I. & Hope, A., NextGen. *Transforming the UK into the world's leading talent hub for the video games and visual effects industries*, Nesta, feb 2011. www.nesta.org.uk/sites/default/files/next_gen_wv.pdf

Luyten, Hans, c.s., *De relatie tussen leerling- en schoolkenmerken en digitale geletterdheid van 14-jarigen: secundaire analyses op de data van ICILS-2013,* Universiteit Twente, aug. 2015. www.kennisnet.nl/uploads/tx_kncontentelements/Rapport_Secundaire_analyses_op_de_Nederlandse_ICILS.pdf

Lye, Sze Yee, et al., 'Exploring the use of online space in an elementary school', In: *Educational Media International,* Volume 49, Issue 3, 2012. www.sciencedirect.com/science/article/pii/S0747563214004634

Lye, Sze Yee en Koh, Joyce Hwee Ling, 'Review on teaching and learning of computational thinking through programming: What is next for K-12?' In: *Computers in Human Behavior, Volume 41, December 2014, Pages 51–61.* www.sciencedirect.com/science/article/pii/S0747563214004634

Naughton, J., 'Why all our kids should be taught how to code'. In: *The Observer,* 2012-03-21 www.theguardian.com/education/2012/mar/31/why-kids-should-be-taught-code

Negroponte, Nicholas. *Being Digital,* Londen, 1995. en.wikipedia.org/wiki/Being_Digital

Nettleford, W., P*rimary School Children Learn to Write Computer Code*. 2013. www.bbc.co.uk/news/uk-england-london-23261504

Papert, S., *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980. dl.acm.org/citation.cfm?id=1095592

Perkovic L., Settle A., Hwang S. and Jones, J. (2010). 'A Framework for Computational Thinking across the Curriculum'. In: *Proceedings of the 2010 Conference on Innovation and Technology in Computer Science Education,* 2010 (p. 123-127). dl.acm.org/purchase.cfm?id=1822126&-CFID=479707165&CFTOKEN=31824809

Peyton Jones, S., *Computing at School: the state of the nation.* UK Computing Research Committee, 2010. www.ukcrc.org.uk

Peyton Jones, S., et al., *Computing at school in the UK,* CACM, april 2013.
research.microsoft.com/en-us/um/people/simonpj/papers/cas/computingatschoolcacm.pdf

Peyton Jones, S., *Teaching Creative Computer Science,* TedxExeter, 2014.
tedxexeter.com/2014/05/06/simon-peyton-jones-teaching-creative-computer-science

Rushkoff, D., Program or be *Programmed: Ten Commands for a Digital Age,* OR Books, New York, 2009.
www.rushkoff.com/program-or-be-programmed

Schmidt, A., et.al. 'Teaching Model-Driven Software Development: Revealing the "Great Miracle" of Code Generation to Students'. In: *Proceedings of the Sixteenth Australasian Computing Education Conference (ACE2014),* Auckland, New Zealand, 2014.
dro.deakin.edu.au/eserv/DU:30065011/craig-unblockingpipeline-2014.pdf

Scott, J., *I love My Smartphone* – Tutor. BCS / Academy of Computing, 2012 www.royalsoced.org.uk/1035_MobileAppDevelopment.html

Scott, J., *I love My Smartphone* – Learner. BCS / Academy of Computing, 2012
www.royalsoced.org.uk/1035_MobileAppDevelopment.html

'Switched On'. Computing at School Newsletter. Summer 2014.
www.computingatschool.org.uk/data/uploads/newsletter-summer-2014.pdf

Teaching Agency, *Subject Knowledge Requirements for Entry into Computer Science Teacher Training* (London, 2012).
www.computingatschool.org.uk/data/uploads/CSSubjectKnowledgeRequirements.pdf

Tolboom, Jos, et al., *Informatica in de bovenbouw havo/vwo,* SLO, april, 2014.
www.slo.nl/downloads/2014/informatica-in-de-bovenbouw-havo-vwo.pdf

Truijens, A., *Onderzoek eerst eens de effecten van ict-onderwijs!*, de Volkskrant, 25 april 2015.
www.volkskrant.nl/opinie/onderzoek-eerst-eens-de-effecten-van-ict-onderwijs~a3980227/

Veen, J., e.a., *SLO-context po,* #11, SLO, jan. 2015.
www.slo.nl/organisatie/slomagazines/context/contextpo

Voorwinden, R., Herman Richter: *'Programmeren ook aanbieden in de kleuterklas',* Kennisnet, januari 2015. www.kennisnet.nl/themas/21st-century-skills/artikelen/blogberichten/programmeren-ook-aanbieden-in-de-kleuterklas

Voorwinden, R., *Maak programmeren een verplicht examenvak in het voortgezet onderwijs,* Kennisnet, januari 2015. www.kennisnet.nl/themas/21st-century-skills/artikelen/blogberichten/programmeren-een-verplicht-examenvak-in-het-voortgezet-onderwijs

Werner, T.H., & Korte, B., *Evaluation of the implementation of the communication of the European Commission E-Skills for the 21st Century, European Commission,* oct 2011. ec.europa.eu/enterprise/sectors/ICT/files/reports/eskills21_final_report_en.pdf

Wing, J., *Research Notebook: Computational Thinking - What and Why?,* Carnegie Mellon, Pittsburgh, 2011. www.cs.cmu.edu/link/research-notebook-computational-thinking-what-andwhy

Zweben, S., *Computing Degree and Enrollment Trends*, Computing Research Association, 2011. cra.org/uploads/documents/resources/taulbee/CS_Degree_and_Enrollment_Trends_2010-11.pdf

Noten

1 Willem Vermeend en Rick van der Ploeg, 'Onderwijs moet op de schop', in: *De Telegraaf*, 16 februari 2015. www.telegraaf.nl/dft/goeroes/rick-willem/23690488/__Onderwijs_moet_op_de_schop__.html

2 Barthel, P., Brock, B. de, Jong, F. de, Lagendijk, I., Smeets, D. en Korbijn, A., *Digitale geletterdheid in het voortgezet onderwijs. Vaardigheden en attitudes voor de 21ste eeuw*. Amsterdam: KNAW, 2012.

3 Zie: www.computable.nl/artikel/nieuws/overheid/5089203/1277202/vak-informatica-moet-op-de-schop.html#ixzz3joVqOvY1.

4 Luyten, Hans, c.s., *De relatie tussen leerling- en schoolkenmerken en digitale geletterdheid van 14-jarigen: secundaire analyses op de data van ICILS-2013,* Universiteit Twente, aug. 2015.

5 "Computational thinking is a way of solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science. Computational thinking involves some familiar concepts, such as problem decomposition, data representation, and modeling, as well as less familiar ideas, such as binary search, recursion, and parallelization." (Jeanette Wing, 2006: *Computational Thinking: A Digital Age Skill for Everyone*)

6 Dredge, S., *Kids coding at school: When you learn computing, you're thinking about thinking*, The Guardian, 22 sept 2014. www.theguardian.com/technology/2014/sep/22/computing-bcs-uk-computing-curriculum

7 Balanskat, A., Engelhardt, K., *Computing our future - Computer programming and coding -Priorities, school curricula and initiatives across Europe*, European Schoolnet, okt 2014. www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887

10 *Hello world!,* The Economist, 20 sept 2014. www.economist.com/news/britain/21618892-training-army-tiny-techies-hello-world

11 Zie: www.education.gov.uk/inthenews/speeches/a00201868/michael-gove-speech-at-the-bett-show-2012

12 Dredge, S., *Kids coding at school: When you learn computing, you're thinking about thinking*, The Guardian, 22 sept 2014. www.theguardian.com/technology/2014/sep/22/computing-bcs-uk-computing-curriculum

14 Lye, Sze Yee en Koh, Joyce Hwee Ling, 'Review on teaching and learning of computational thinking through programming: What is next for K-12?' In: *Computers in Human Behavior*, Volume 41, December 2014, Pages 51–61. www.sciencedirect.com/science/article/pii/S0747563214004634

15 Zie: tedxexeter.com/2014/05/06/simon-peyton-jones-teaching-creative-computer-science

16 Berry, M., *Computing in the national curriculum*. A guide for primary teachers. CAS/Naace, 2013. www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf

17 Meer informatie over dit hulpmiddel: www.360safe.org.uk

18 Zie ook: knutsfordcomputing.com/welcome-knutsford-academy-computing-learning-centre

19 Berry, M., *Computing in the national curriculum - a guide for primary teachers*, Naace / Computing at School, 2013, p.19

20 Zie onder andere: Brown, N., Sentance, S., Crick, T., and Humphreys, S. 2014. 'Restart: The Resurgence of Computer Science in UK Schools.' In: *ACM Transactions on Computing Education* (TOCE), Volume 14 Issue 2, New York, June 2014, Art. 9, p. 1-22

21 Zie o.a.: codeclubpro.org

22 Kemp, P., *Computing in the national curriculum - a guide for secondary teachers*, Naace / Computing at School, 2013, p. 26

23 Berry, M., *QuickStart Computing; a CPD toolkit for secondary teachers*, Computing at School, 2014

24 Zie: www.computingatschool.org.uk/data/uploads/CASInfrastructure.pdf

25 *Teaching creative computer science: Simon Peyton Jones at TEDxExeter*, YouTube: www.youtube.com/watch?v=Ia55clAtdMs

26 "[...] zeker 3 en hoogstwaarschijnlijk meer dan 5 jaar" is gebaseerd op de ervaringen van de Britse docenten die reageerden op onze enquête. Het computing-curriculum werd ingevoerd: drie jaar vóór het verplicht werd. Die tijd was in ieder geval nodig. In de praktijk lijkt er zeker nog twee jaar extra nodig te zijn.

27 www.kennisnet.nl/artikel/schoolbesturen-ontwikkelen-leerlijn-programmeren-voor-basisonderwijs/

28 techniekpact.nl/

29 www.platformbetatechniek.nl/publicaties/advies-denktank2032

30 www.vhto.nl/fileadmin/user_upload/documents/publicaties/Meisjes/VHTO_KiesInformatica_LR.pdf

31 Zoals: Astrid Poot (www.lekkersamenklooien.nl), en Ihsane Beyd (docente Frans, die met haar leerlingen lesjes Frans programmeert). Zie ook: www.kennisnet.nl/artikel/programmeren-als-onderdeel-van-de-franse-les

32 Brennan, K., & Resnick, M. *New frameworks for studying and assessing the development of computational thinking*. American Educational Research Association meeting, Vancouver, BC, Canada, 2012

33 www.onderwijsmaakjesamen.nl/actueel/werken-met-portfolios/

34 *Mediawijsheid op de basisschool - Succesverhalen van 21 leerkrachten*, Kennisnet & MKO, 2013. archief.kennisnet.nl/fileadmin/contentelementen/kennisnet/Dossier_mediawijsheid/Boek_Mediawijsheid_op_de_basisschool.pdf

35 www.kennisnet.nl/artikel/programmeren-als-onderdeel-van-de-franse-les/

36 Zie: doorbraakonderwijsenict.nl

37 Zie: www.computerweekly.com/news/4500253068/One-third-of-schools-admit-to-making-no-investment-in-coding-training-for-teachers

38 Tolboom, Jos, et al., *Informatica in de bovenbouw havo/vwo*, SLO, april, 2014. www.slo.nl/downloads/2014/informatica-in-de-bovenbouw-havo-vwo.pdf

39 Bron: www.waag.org/nl/project/teacher-maker-camp

40 Tolboom, Jos, et al., *Informatica in de bovenbouw havo/vwo*, SLO, april, 2014. www.slo.nl/downloads/2014/informatica-in-de-bovenbouw-havo-vwo.pdf

41 Fisser, P., Hoeven, M. v.d., Thijs, A., *Digitale geletterdheid en 21e eeuwse vaardigheden in het funderend onderwijs: een conceptueel kader*, SLO, jan. 2014.

42 Jeanette Wing, 2006: *Computational Thinking: A Digital Age Skill for Everyone.*

43 Zie: www.computingatschool.org.uk/data/uploads/newsletter-summer-2014.pdf

Computing-onderwijs in de praktijk

Deze publicatie delen? Kies uit één van onderstaande sociale media.

Deel via Twitter

Deel via LinkedIn

Deel via Facebook

Colofon

Computing-onderwijs (Wat kunnen we leren van de Britten?) is een uitgave van Kennisnet.

Realisatie
- Remco Pijpers – onderzoek en coördinatie
- Louis Stiller – onderzoek en tekst
- Henk Boeke – eindredactie
- Optima Forma bv – vormgeving
- Fotografie – Etienne Oldeman (p.3, p.13, p.48), Rodney Kersten (p.35), Fotolia (omslag)

Stichting Kennisnet
Kennisnet laat ict werken voor het onderwijs en is de publieke ict-partner voor het onderwijs. Met expertise, voorzieningen en innovatie brengt zij het onderwijs in positie om maximale kracht uit ict te halen.
- e-mail: info@kennisnet.nl
- website: www.kennisnet.nl
- contactpersoon: Remco Pijpers r.pijpers@kennisnet.nl

Adviezen
- Allard Strijker – leerplanontwikkelaar ict, SLO
- Don Zuiderman – docent ict & Onderwijs, Pabo Hogeschool Utrecht
- Guus Wijngaards – voormalig lector eLearning, Hogeschool InHolland
- Jan Lepeltak – voormalig lector ict en veranderende didactiek aan de Noordelijke Hogeschool Leeuwarden
- Sjoerd de Vries – onderwijsmanager/ onderzoeker Toegepaste Psychologie, Saxion Hogeschool
- Sandra Legters – coördinator Programmeertalen en Techniek, Stichting Openbaar Primair Onderwijs NoordOost Achterhoek
- Teun Meijer – coördinerend ict'er onderwijsgroep Fier, en coördinator SD-Fryslân
- Nicole de Groot – coördinator, schoolbestuur Kits Primair, Drenthe
- Ihsane Beyd – docente Frans, Jan van Brabant-college, Helmond
- Pieter Paul Eppings – leerkracht, Vlissingse schoolvereniging
- Harriet Leget – projectleider, Stichting Kennisnet
- Wietse van Bruggen – specialist 'maker education', Stichting Kennisnet

Kennisnet laat ict werken voor het onderwijs en is de publieke ict-partner voor het onderwijs. Met expertise, voorzieningen en innovatie brengt zij het onderwijs in positie om maximale kracht uit ict te halen.

Stichting Kennisnet
Paletsingel 32
2718 NT Zoetermeer
Postbus 778
2700 AT Zoetermeer

T 0800 321 22 33
E info@kennisnet.nl
I kennisnet.nl