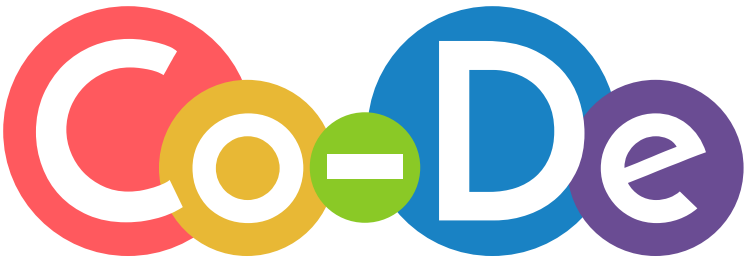
KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN



Co-De: een digitaal leerplatform voor computationeel denken

Zimcke Van de Staey
Tobias Verlinde

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdoptie artificiële intelligentie

Promotoren:

Prof. dr. Bart Demoen
Prof. dr. Bern Martens

Assessoren:

Prof. dr. ir. Yolande Berbers
Prof. dr. ir. Hendrik Blockeel

Academiejaar 2017 – 2018

Voorwoord

> "*Within just a few decades, computational thinking has changed the way we all live, work and play. It has changed the way science is done too; it has won wars; created whole new industries and saved lives.* [12, p. v]"
> – Paul Curzon, oprichter *cs4fn*

Het onderzoeksproject was een bijzonder leerrijke en boeiende ervaring voor ons beiden. Van onze gedurfde keuze om te werken rond een educatief thema binnen een masterproef computerwetenschappen, hebben we geen minuut spijt gehad doorheen het jaar. Wij zijn ervan overtuigd dat zonder onze technische kennis Co-De er heel anders zou hebben uitgezien en de toekomstkansen van het leerplatform minder groot zouden zijn. We zijn dan ook bijzonder trots op 'ons' leerplatform Co-De en onze kleine bijdrage aan beter informatica-onderwijs. In de toekomst willen wij ons blijven inzetten voor Co-De en we hopen dat het afronden van deze masterproef niet het einde van het leerplatform betekent.

We willen ook enkele mensen uitdrukkelijk bedanken voor hun bijdrage aan dit project. In de eerste plaats zijn wij onze promotor prof. dr. Bart Demoen en co-promotor prof. dr. Bern Martens enorm dankbaar. Ons bijna wekelijks overleg was steeds zeer aangenaam en constructief, maar daarnaast was er ook regelmatig ruimte voor een grapje en andere gespreksthema's. Daarnaast willen we graag onze assessoren, prof. dr. ir. Hendrik Blockeel en prof. dr. ir. Yolande Berbers, bedanken voor het lezen van deze masterproef. Voor technische ondersteuning konden we herhaaldelijk een beroep doen op de medewerkers van het departement computerwetenschappen van de KU Leuven. In het bijzonder willen we Bart Swennen en Greg Vanhove bedanken voor hun hulp bij het opzetten en het onderhouden van Co-De. Ook onze partners, Alexander Gim Ho Tang en Leen Hertens, verdienen een speciale vermelding en zijn we enorm dankbaar. Niet alleen hebben zij ons gedurende het hele project gesteund, zij hebben zich ook herhaaldelijk bereidwillig getoond om onderdelen te testen, te luisteren, tekst na te lezen en nog zo veel meer. Onze ouders en de rest van onze familie stonden het afgelopen jaar steeds voor ons klaar om ons te steunen bij dit project. Ook aan hen een welgemeende dankjewel daarvoor. We willen ook graag Annelies Claessens, Mathias De Baets en Mieke Leroy bedanken voor de talloze uren die zij staken in het nalezen van deze tekst en de gebruikershandleiding. Tot slot zijn we ook alle testgebruikers, gebruikers en geïnteresseerden enorm dankbaar voor hun tijd en feedback waarmee ze ons hielpen om Co-De uit te bouwen.

Tobias & Zimcke

Inhoudsopgave

Samenvatting

Co-De is een digitaal leerplatform voor computationeel denken dat online toegankelijk is via https://code.experiments.cs.kuleuven.be. De digitale wereld waarin we leven bepaalt mee welke basisvaardigheden onontbeerlijk zijn. Menig onderzoeker is het erover eens dat computationeel denken (CD) vandaag even belangrijk is geworden als lezen en schrijven. Verscheidene studies tonen ook aan dat CD een belangrijke rol moet spelen in de technologische opvoeding van jongeren. In Vlaanderen staat de integratie van computationeel denken in het secundair onderwijs nog in de startblokken. Dankzij Co-De is er voor het eerst een integraal Nederlandstalig platform voor scholen, met lesmateriaal over CD en met een uitgebreide gebruikershandleiding. Het lesmateriaal bestaat uit een lessenreeks van vier lessen en een korte wegwijzer over computationeel denken. De meeste lessen bestrijken enkele (één tot vier) lesuren. Elke les focust op een specifiek onderwerp en brengt één of meerdere informaticaconcepten aan, zoals *gerichte en ongerichte grafen*, *dynamisch programmeren*, *zoekalgoritmes of heuristieken*, *toestandsautomaten*... De gebruikers worden bij het volgen van een les telkens bevraagd en nadien geïnformeerd over hun gebruik van CD. De componenten van CD die het platform behandelt, zijn dezelfde als de elementen die naar voren worden geschoven in het Brits onderwijs waar CD reeds deel uitmaakt van de leerplannen, namelijk: *abstractie*, *veralgemening*, *decompositie*, *algoritmisch denken* en *evaluatie*. Het leerplatform zelf is gebaseerd op de *open source* leeromgeving Moodle en wordt door middel van Docker gedeployed op een server van de KU Leuven.

Deze masterproef onderzoekt de tendensen rond computationeel denken zowel in het algemeen als binnen het onderwijs en definieert een eigen opvatting van het begrip. We gaan ook op zoek naar ondersteunende modellen waarop we het leerplatform en de lesinhoud kunnen baseren. Daarnaast bestudeert deze masterproef ook de (technische) vereisten verbonden aan een online leerplatform voor CD en de mogelijkheden om deze te realiseren. De vraag hoe de functionaliteit van Moodle kan worden gewijzigd en aangevuld om te voldoen aan de vereisten van Co-De staat daarbij centraal. Een essentieel aspect is dat leerkrachten op een gebruiksvriendelijke manier het leerpad van de lessen kunnen aanpassen. Ook de uitbreidbaarheid van het platform is tevens belangrijk. Zowel op functioneel vlak als op vlak van lesinhoud is er immers nog heel wat groei mogelijk.

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen en symbolen

Afkortingen

`$code`	https://code.experiments.cs.kuleuven.be De portaalpagina van het leerplatform.
[koː-dɛ]	Fonetische notatie van Co-De volgens het IPA.
CAS	*Computing At School* De Britse organisatie voor computerwetenschappen op school.
CD	Computationeel denken.
CT	*Computational thinking* of computationeel denken.
e-learning	*Electronic learning* of digitaal leren.
K-12	*K twelve* of de verzameling van het basis- en secundair onderwijs (gemiddeld 4 t.e.m. 19 jaar) in de Verenigde Staten.
KVAB	Koninklijke Vlaamse Academie van België voor Wetenschappen en Kunsten.
MOODLE	*Modular Object-Oriented Dynamic Learning Environment* Digitale *open source* leeromgeving, Moodle.
STEM	*Science, Technology, Engineering and Mathematics* of wetenschappen, technologie, ingenieurstechnieken en wiskunde.
WiPSCE	*Workshop in Primary and Secondary Computing Education* Een conferentie over informatica-onderwijs in het lager- en secundair onderwijs.

Symbolen

Hoofdstuk 1

Inleiding

Computationeel denken is een begrip dat de laatste jaren enorm aan belang wint. Het omvat de algemene vaardigheden die nodig zijn om de (complexe) problemen van tegenwoordig te begrijpen en op te lossen. Verschillende auteurs geven aan dat computationeel denken (CD) even belangrijk is als elementaire vaardigheden zoals lezen, schrijven en rekenen.

> *We are preparing young people for jobs that don't yet exist, requiring technologies that have not yet been invented, to solve problems of which we are not yet aware. Skills and technologies may seem topical, but quickly go out of date; principles and ideas [CT] may appear less up-to-the-minute but are transferrable and remain important and applicable a decade or two later* [17, p. 3].
>
> Simon Peyton Jones (*CAS: the state of the nation*, 2009)

Het valt dan ook niet te verbazen dat heel wat studies aantonen dat computationeel denken een belangrijke rol moet spelen in de technologische opvoeding van jongeren. In enkele van onze buurlanden is CD al opgenomen in het informatica-onderwijs of is men daar intensief mee bezig. In Vlaanderen[1] heeft CD echter nog geen vaste plaats gekregen binnen het onderwijs.

> *Door het ontwikkelen van hun creativiteit, het verschaffen van inzicht en het aanleren van computationeel denken, worden kinderen bewust van de mogelijkheden en gevaren van de geïnformatiseerde wereld* [15].
>
> Giselle Vercauteren (*Datanews - Knack*, 2015)

Om computationeel denken te kunnen aanleren, is lesmateriaal noodzakelijk. In het Vlaams onderwijs dient dit lesmateriaal beschikbaar te zijn in het Nederlands. Computationeel denken aanleren d.m.v. een tekst- of werkboek kan zeker, maar die aanpak mist de nauwe band tussen CD en de hedendaagse digitale realiteit.

[1] Onderwijs is in België een bevoegdheid van de gemeenschappen.

Daarenboven valt onder CD ook leren programmeren[2] en dat kan men onmogelijk ten gronde aanleren op papier[3]. Een andere, dynamische en interactieve leervorm dringt zich dus op. Wij bouwden daarom een digitaal leerplatform waarop computationeel denken kan worden aangeleerd. Op zo een leerplatform kan het lesmateriaal op een multimediale manier worden aangereikt. We dopen ons leerplatform "Co-De" [koː-dɛ][4], verwijzend naar "Computationeel Denken".

De intentie van deze masterproef is om Co-De op te bouwen en vorm te geven, met als eindpunt een platform waarop leerlingen zelfstandig kunnen werken binnen een klascontext om te leren computationeel denken. We hebben als doel om een lessenreeks uit te werken op het platform met voldoende lesmateriaal voor één wekelijks lesuur informatica gedurende één schooljaar. Het lesmateriaal en het platform moeten volledig in het Nederlands beschikbaar zijn zodat het geschikt is voor het Vlaams onderwijs.

Er zijn rond CD al een aantal (werk)boeken te vinden met lesinhoud en oefeningen die verwijzen naar websites, apps en programma's waarmee gewerkt kan worden. Een interactieve leeromgeving waarin een scala aan oefeningen en evaluatiemethodes gecentraliseerd wordt aangeboden, bestaat nog niet (in het Nederlands). Het is duidelijk dat daarop plaats moet zijn voor tekst en uitleg, afbeeldingen, filmpjes en andere mediabronnen, maar ook voor *unplugged* oefeningen, online opdrachten, vraagstellingen en programmeeropdrachten. Voor deze masterproef wordt zo'n leeromgeving opgezet en opgevuld met een lessenreeks. We onderzoeken hoe de verschillende aspecten van CD aan bod kunnen komen in de lessen. Daarom dient computationeel denken eerst te worden gedefinieerd en gekaderd.

Het doel van het leerplatform is om het te kunnen gebruiken op middelbare scholen in de lessen gerelateerd aan informatica of STEM. Enerzijds heeft het platform als doelstelling om leerkrachten zoveel mogelijk ondersteuning te bieden bij hun taak als pedagoog. Deze masterproef onderzoekt daarom in welke vorm de leerstof op het platform het meest geschikt is. Anderzijds moet het platform leerkrachten toestaan om de lessen aan te passen op een gebruiksvriendelijke manier. Dit is nodig aangezien we de lessen uitbouwen voor een breed publiek, maar elke klasgroep anders is. In het algemeen richten we ons binnen de derde graad van het secundair onderwijs tot richtingen waarin van de leerlingen een bepaald abstractieniveau verwacht wordt. Veel van het (les)materiaal zal echter ook bruikbaar zijn voor andere leeftijds- en doelgroepen.

Tot slot moet het platform klaar zijn om uitgebreid te worden. Zowel op functioneel vlak als op het vlak van de aangeboden lessen. De opbouw, de concrete invulling en het gebruik moeten daarom ook zorgvuldig worden gedocumenteerd.

[2] CD omvat meer dan enkel leren programmeren, zie hoofdstuk 2.

[3] Al kan men veel programmeervaardigheden en -concepten ook *unplugged* aanleren. Zie ook: "Deze computerwetenschapper doet een voorzet voor écht programmeeronderwijs [34]".

[4] Volgens het internationaal fonetisch alfabet (IPA).

Om een leerplatform en lessen rond CD te kunnen bouwen is het allereerst belangrijk om uit te zoeken wat CD juist betekent. Daarom starten we met een hoofdstuk dat zich toespitst op het begrip computationeel denken (hoofdstuk 2 vanaf p. 5). Eerst wordt de literatuur grondig besproken. De oorsprong en het debat rond de concrete invulling van het paradigma CD staan daarin centraal, maar er is ook ruimte voor praktijkvoorbeelden en een vergelijking met probleemoplossend denken. Vanuit deze literatuur zullen we een eigen opvatting van computationeel denken vormgeven aan de hand van vijf kernelementen: *abstractie*, *veralgemening*, *decompositie*, *algoritmisch denken* en *evaluatie*. Deze visie zullen we vervolgens systematisch meenemen bij het ontwikkelen van het leerplatform en de lesinhoud. In dit hoofdstuk analyseren we ook het belang van CD in het algemeen en meer specifiek in het (secundair) onderwijs.

Na de uiteenzetting over computationeel denken wordt het leerplatform gepresenteerd (hoofdstuk 3 vanaf p. 25). In dit hoofdstuk is heel wat aandacht voor de technische aspecten van het leerplatform zoals de gekozen technologie (Moodle), hoe het platform werd gedeployed (via Docker) en welk technisch onderhoud vereist is. Aangezien er niet veel literatuur bestaat over het software-ontwerp en de software-architectuur van educatieve leeromgevingen is dit geen eenvoudige kwestie. De verschillende technische en inhoudelijke keuzes worden zo goed als mogelijk gestaafd en, waar mogelijk, vergeleken met alternatieven. Daarnaast worden ook de verschillende functionaliteiten van Co-De toegelicht. Zowel de technische opbouw, het beheer en het gebruik komen daarbij aan bod. De verschillende (technische) aspecten van het leerplatform worden ook individueel getest en bijgestuurd indien nodig.

Eens de opbouw van het platform is geschetst, worden ook de lessen uitgebreid toegelicht in een apart hoofdstuk (hoofdstuk 4 vanaf p. 69). Eerst is er ruimte voor onderliggende (didactische) modellen zoals het 4C/ID-model en vier eigenschappen die worden toegewezen aan de lesonderdelen. Ook het ontwikkelproces van de lessen wordt uit de doeken gedaan. Ten tweede worden de uitgewerkte lessen gepresenteerd en geanalyseerd. We nemen de lessen één voor één onder de loep en schetsen de probleemstelling, lichten het standaard leerpad toe en koppelen de les aan enkele kernelementen van CD. Bovendien worden ook de keuzes bij het ontwerp van elke les toegelicht.

In de nabeschouwing (hoofdstuk 5 vanaf p. 101) blikken we terug op het bereikte resultaat en kijken we vooruit naar de toekomst van Co-De. We trachten om de verschillende aspecten op te sommen die de toekomst van Co-De mee zullen bepalen. Het technische onderhoud speelt hierin zeker een belangrijke rol. In de nabeschouwing is ook ruimte voor reflectie en worden enkele ideeën opgesomd voor de verdere ontwikkeling van Co-De.

Hoofdstuk 2

Computationeel denken

Een belangrijk element van (informatica)wetenschappen is computationeel denken. Dit is een concept dat de laatste jaren meer en meer aan belang wint, ook in Vlaanderen. Computationeel denken (CD) wordt vermeld in zowat elke discussie over informatica- en STEM-onderwijs.

Dit hoofdstuk bevat ten eerste een korte literatuurstudie over computationeel denken. Daarin wordt het ontstaan van het begrip besproken en worden enkele theoretische interpretaties naast elkaar geplaatst. Er is namelijk nog geen universele consensus over de precieze inhoud van computationeel denken. Daarna volgt er een meer praktijkgerichte benadering en wordt het belang van CD in het Vlaams onderwijs gekaderd. Door verschillende opvattingen uit de praktijk te beschouwen, wordt een invulling van CD voor Co-De afgeleid. Er wordt ook een korte vergelijking gemaakt tussen computationeel denken en probleemoplossend denken. Dit is nodig omdat computationeel en probleemoplossend denken nauw verwant zijn, maar er toch ook een aantal belangrijke verschillen bestaan.
Ten tweede wordt in dit hoofdstuk besproken hoe werken rond CD een goede invulling kan zijn voor de lessen informatica, in het bijzonder in het secundair onderwijs.
Ten derde worden vijf elementen van computationeel denken apart uitgelegd. De tekst beperkt zich hier tot de elementen die centraal staan op het platform Co-De.
Het hoofdstuk sluit af met een kort overzicht van andere aspecten van computationeel denken en met een kort besluit.

2.1 Literatuurstudie

2.1.1 Ontstaan van computationeel denken

In 2006 verscheen het artikel "Computational Thinking" van de Amerikaans onderzoekster Jeannette Wing [35]. Dit artikel is niet de eerste publicatie die de term computationeel denken vermeldt, dat gebeurde al in 1980 door Seymour Papert [23]. De definitie die Wing formuleert, wordt wel algemeen beschouwd als de eerste academische definitie van computationeel denken. Ze definieert computationeel denken (CD) als volgt:

> *Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation* [35, p. 33].

Computationeel denken is, volgens Wing, een verzamelnaam voor verschillende technieken die toegepast kunnen worden om complexe problemen op te lossen. Welke technieken precies onder CD vallen om de lading volledig te dekken, bespreekt Wing niet in detail. Ze noemt wel een reeks componenten en voorbeelden op die volgens haar zeker thuishoren in CD. De lijst is te lang om hier volledig op te nemen, maar de belangrijkste componenten zijn: *evaluatie*, *recursiviteit*, *representatie*, *duidelijkheid en eenvoud*, *abstractie*, *decompositie*, *verificatie*, *modulariteit*, *heuristieken* en *algoritmisch denken* [35]. Ook leren programmeren maakt volgens Wing deel uit van computationeel denken, maar CD mag hier zeker niet toe herleid worden [35, p. 35].

Wing ziet computationeel denken als een reeks vaardigheden die elk individu zou moeten beheersen, naast andere basisvaardigheden zoals rekenen, lezen en schrijven:

> *This kind of thinking will be part of the skill set of not only other scientists but of everyone else. Ubiquitous computing is to today as computational thinking is to tomorrow. Ubiquitous computing was yesterday's dream that became today's reality; computational thinking is tomorrow's reality* [35, p. 34].

Het is vanaf het ontstaan van het begrip duidelijk dat computationeel denken meer en meer impact zal uitoefenen op het dagelijks leven. Iedereen zal aan de slag moeten gaan met de bouwstenen van computationeel denken om te kunnen omgaan met hedendaagse vraagstukken.

2.1.2 Debat

Wings korte uiteenzetting over computationeel denken laat ruimte voor verschillende interpretaties van CD. Welke vaardigheden of attitudes prioriteit hebben binnen computationeel denken, staat bijvoorbeeld ter discussie. Haar initieel werk rond CD geeft ook geen uitleg over hoe de vaardigheden kunnen worden getraind. Sinds Wing haar artikel publiceerde, hebben veel onderzoekers en organisaties getracht om haar inzichten te herdefiniëren.

De Europese Commissie publiceerde in 2016 een rapport over de integratie van CD in het verplicht leerprogramma van het leerplichtonderwijs [3]. Daarin wordt eerst een synthese gemaakt van de elementen van CD zoals die in Wings artikel vervat zitten en de elementen die in vaak geciteerde literatuur aan bod komen. De kernelementen van CD die daardoor geïdentificeerd konden worden zijn opgesomd in tabel 2.1. Deze opsplitsing in zes elementen is veel concreter dan de algemene definitie van Wing.

Element van CD	**Beschrijving**
Abstractie (*Abstraction*)	Onnodige details achterwege laten; een goede representatie kiezen van het probleem
Algoritmisch denken (*Algorithmic thinking*)	Een stappenplan definiëren
Automatisering (*Automation*)	Een computer een reeks repetitieve taken laten uitvoeren
Decompositie (*Decomposition*)	Onderdelen van het probleem afzonderlijk oplossen
Debuggen (*Debugging*)	Systematische evaluatie van de resultaten door o.a. testen en logisch denken
Veralgemening (*Generalization*)	Vergelijken met gerelateerde problemen; patroonherkenning; nieuwe problemen oplossen met dezelfde oplossingsmethode

Tabel 2.1: De kernelementen van computationeel denken volgens het recent rapport van de Europese Commissie [3, p. 18]. De beschrijving van elk element volgt de bewoordingen uit dit rapport.

In "Communications of the ACM" verscheen in juni 2017 een artikel van Peter J. Denning dat de verschillende invullingen van CD door enkele toonaangevende organisaties naast elkaar plaatst [14]. Tabel 2.2 toont een herwerkt overzicht van die studie. Vergelijkbare eigenschappen bij de verschillende organisaties staan op dezelfde hoogte naast elkaar in de tabel. In de eerste kolom staat de interpretatie van de *Computer Science Teachers Association* (*CSTA*)[1], een Amerikaanse organisatie die

[1] http://www.csteachers.org/.

actief is sinds 2004 [10]. Ze houdt zich bezig met de ondersteuning van leerkrachten informatica uit het *K12*-onderwijs. *K12*-onderwijs duidt ruwweg op het onderwijs voor 4- t.e.m. 19-jarigen. Daarnaast staat de opvatting van *Computing at School* (*CAS*), de Britse organisatie voor computerwetenschappen op school[2]. Ten slotte is er nog de interpretatie van de internationale beweging *ISTE* of *International Society for Technology in Education*[3]. Die beweging probeert wereldwijd leerkrachten en deskundigen met elkaar in contact te brengen, ze telt intussen meer dan 16 000 online leden[4]. In volgorde van dalende frequentie komen de volgende componenten het vaakst voor: *abstractie*, *algoritmisch denken*, *evaluatie*, *data-analyse*, *veralgemening*, *datarepresentatie* en *decompositie*. Vijf elementen hiervan zijn ook terug te vinden in de vorige tabel (tabel 2.1), namelijk *abstractie*, *algoritmisch denken*, *automatisatie*, *decompositie* en *veralgemening*.

Computer Science Teachers Association (CSTA)	**Computing at School (CAS)**	**International Society for Technology in Education (ISTE)**
Problemen formuleren		Oplossingen ontwerpen
	Logisch redeneren	
Data-analyse		Dataverzameling en -analyse
Abstractie (modellen en simulatie)	Abstractie	Abstractie, simulatie
Algoritmisch denken	Algoritmisch denken	Algoritmen
Evalueren (correctheid en efficiëntie)	Evaluatie	Testen
Veralgemening	Veralgemening	
	Patronen	
	Voorstelling	Datavoorstelling
	Decompositie	Decompositie
		Automatiseren
		Parallelliseren

Tabel 2.2: Vergelijking tussen de vaardigheden die deel uitmaken van computationeel denken bij verschillende toonaangevende instanties. Gebaseerd op een studie van P.J. Denning [14, p. 34].

[2] https://www.computingatschool.org.uk/.
[3] https://www.iste.org/.
[4] https://www.iste.org/docs/pdfs/iste-annual-report-2017.pdf.

Sommige auteurs delen de hoog-niveau elementen van CD nog verder op. Hoskey en Zhang onderscheiden in hun werk zestien concrete vaardigheden binnen computationeel denken [16]. Door meer in detail te gaan, willen ze het onderscheid tussen computationeel denken en andere vormen van denken duidelijker schetsen. De vaardigheden die aan bod komen, zijn gekozen omdat ze specifiek voor het informatica-onderwijs van toepassing zijn. De zestien elementen zijn door Hoskey en Zhang onderverdeeld in twee niveaus, namelijk laag en hoog, zoals weergegeven in tabel 2.3. De twee niveaus hebben betrekking tot de manier waarop met een computer gewerkt wordt. De vaardigheden op laag niveau gaan over het implementeren en coderen van een concrete oplossing. Op hoog niveau gaan de vaardigheden meer over het ontwerpen van een algemene oplossingsmethode. Individuele begrippen uit de tabel zijn interessant wanneer omvangrijke componenten van CD nood hebben aan een preciezere invulling. Zo zitten *iterative thinking*, *recursive thinking* en *parallel thinking* (tabel 2.3) bijvoorbeeld vervat in het *algoritmisch denken* uit eerder genoemde bronnen.

Laag niveau Implementeren en coderen	**Hoog niveau** Ontwerpen, modelleren, probleemoplossend denken
Absolute vs relative thinking	Abstraction
Automation thinking	Object-oriented thinking
Categorization	Parallel thinking
Compound thinking	Recursive thinking
Information representation	Scalability thinking
Interface thinking	
Iterative thinking	
Modularization	
Reuse thinking	
Sequential thinking	
Unique identification	

Tabel 2.3: De verschillende vaardigheden omvat door het paradigma computationeel denken volgens Hoskey en Zhang [16].

De verschillende tabellen maken duidelijk dat er meerdere interpretaties van computationeel denken naast elkaar bestaan. Het is de normale gang van zaken om computationeel denken te classificeren in verschillende componenten. Die classificatie gebeurt niet altijd op hetzelfde niveau. Sommige auteurs bestempelen *algoritmisch denken* als kernelement van CD terwijl anderen concretere vaardigheden als *iteratief, recursief en parallel denken* onderscheiden. Bovendien gebruikt niet iedereen dezelfde grootteorde, namelijk een onderverdeling in vijf elementen is minder verfijnd dan een onderverdeling in zestien elementen. Het is wel zo dat een aantal elementen van computationeel denken vaker voorkomen dan andere. Het zijn die elementen die op termijn misschien aanleiding kunnen geven tot een eenduidige definitie.

2.1.3 Computationeel denken in de praktijk

Hoewel de literatuur aangeeft dat er nog onduidelijkheid is over welke componenten al dan niet deel uitmaken van CD en welke prioritair zijn, wordt er in de praktijk minder gevarieerd in de aspecten die aan bod komen. Hieronder bekijken we een aantal organisaties en initiatieven die het ontwikkelen van ons leerplatform kunnen inspireren of ondersteunen. Deze voorbeelden zijn geografisch dicht bij Vlaanderen gesitueerd en hebben een sterke band met het onderwijs.

Groot-Brittannië wordt beschouwd als het eerste Europese land waar computationeel denken deel ging uitmaken van het leerplichtonderwijs. Daar gaat mee gepaard dat er een aantal organisaties actief zijn die het belang van computationeel denken proberen in de verf te zetten. De organisatie *cs4fn* (*Computer Science for Fun*), opgericht door Paul Curzon en Peter McOwan, publiceert sinds 2005 met regelmaat uitgewerkte materialen voor leerkrachten en jongeren rond computerwetenschappen[5]. Die publicaties tonen vaak hoe en welke elementen van CD verwerkt zijn in de opdrachten. De nadruk ligt daar op: *algoritmisch denken*, *abstractie*, *veralgemening*, *patroonherkenning*, *representatie*, *decompositie*, *evaluatie* en *logisch denken*. In de *Guide for Teachers* [9, 7] die *Computing at School* (*CAS*) in 2014 publiceerde, wordt het aantal aspecten van computationeel denken nog meer beperkt (ook ten opzichte van *CAS* uit tabel 2.2). Volgens de handleiding ligt de nadruk op *algoritmisch denken*, *decompositie*, *veralgemening*, *abstractie* en *evaluatie* [7]. Het zijn deze aspecten die actief worden aangeleerd in het Brits onderwijs.

Een ander initiatief uit de praktijk is de *Bebras Computing Challenge* [2]. Dit is een internationale wedstrijd die tot doel heeft om leerlingen te laten kennismaken met CD en ze hierin te sterken. In 2017 werd de Bebras-wedstrijd in meer dan dertig landen georganiseerd, waaronder België [2]. Deelnemers van tien tot en met achttien jaar zijn welkom. Er is geen vereiste inhoudelijke achtergrond voor de deelnemers en deelname is mogelijk na registratie van de leerling door één van zijn leerkrachten. Elke Bebras-opgave bestaat uit een tekstuele beschrijving van een probleem, waarna een mogelijke situatie en verloop van het probleem meestal visueel wordt toegelicht. Men verwacht dat elke opgave oplosbaar is binnen de drie minuten [2]. Na meer dan tien jaar is er genoeg materiaal om de inhoud van Bebras te evalueren. Dagiene en Sentance [13] analyseerden de opgaven van 2015, om na te gaan welke aspecten van CD aan bod kwamen. Bovendien wilden ze aantonen dat de opgaven ook hun weg zouden kunnen vinden naar een verplicht leerprogramma, omdat ze verschillende zinvolle vaardigheden trainen. De organisatie van Bebras preciseert zelf niet welke elementen van CD vervat zitten in hun opgaven. Onderzoekers Dagiene en Sentance nemen in hun analyse vijf vaardigheden op, gebaseerd op het model van *CAS*: *abstractie*, *decompositie*, *algoritmisch denken*, *evaluatie* en *veralgemening* [13]. Die vaardigheden werden zo gekozen omdat ze de verschillende stappen uit het probleemoplossend proces belichamen.

[5] Een groot deel van die publicaties zijn beschikbaar op http://www.cs4fn.org. Een voorbeeld is een boekje met drie puzzels, waaronder de paardenronde en stadsgids: https://cs4fndownloads.files.wordpress.com/2016/02/puzzlingtours-booklet.pdf.

Als laatste initiatief bekijken we “Zo denkt een computer” [1], een uitgave voor het Vlaams onderwijs uitgebracht in het kader van het Codefestival 2017[6]. Het doel van die publicatie is om een praktische gids aan te bieden waar scholen zelf mee aan de slag kunnen gaan. Daarin staan zowel theoretische bevindingen als praktijkvoorbeelden om mee te experimenteren. De elementen van CD die opgesomd worden zijn: *algoritmen*, *decompositie*, *patroonherkenning* en *abstractie*.

Tabel 2.4 vat de verzameling kernelementen van CD samen voor de besproken visies uit de praktijk. *Algoritmisch denken*, *abstractie* en *decompositie* komen daarin bij alle instanties voor. Daarnaast zijn ook *veralgemening* en *evaluatie* duidelijk belangrijk. Deze vijf elementen zullen ook op Co-De een prominente plaats innemen.

	cs4fn	**CAS** Elementen uit tabel 2.2	**CAS** Guide for Teachers	**Bebrasopgaven** Analyse Dagiene en Sentance	**Zo denkt een computer**
Abstractie	+	+	+	+	+
Algoritmisch denken	+	+	+	+	+
Decompositie	+	+	+	+	+
Evaluatie	+	+	+	+	
Veralgemening	+	+	+	+	
Patroonherkenning	+	+			+
Logisch denken	+	+			
Representatie	+	+			

Tabel 2.4: Een vergelijking van de kernelementen van computationeel denken bij verschillende organisaties of initiatieven uit de praktijk geordend volgens dalende frequentie.

[6] Meer informatie over het Codefestival via http://codeweek.eu/ of https://www.klascement.net/kiezenvoorstem/focus/Programmeren.

2.1.4 Het belang van computationeel denken

In Vlaanderen wint computationeel denken stilaan aan belang als noodzakelijk denkvermogen om de digitale evolutie te kunnen volgen. Men zou dan ook verwachten dat informaticawetenschappen een solide plaats hebben ingenomen in het onderwijs. Informatica in het Vlaams leerplichtonderwijs is op dit moment echter zeer versnipperd over de verschillende koepels en richtingen. Bovendien is de lesinhoud vaak afgestemd op het leren gebruiken van bepaalde software en computers, maar komt het leren computationeel denken nog maar in beperkte mate aan bod.

De groeiende interesse voor leren computationeel denken is onder andere te danken aan de initiatieven uit het Verenigd Koninkrijk en Nederland[7] waar een duidelijk beleid over informaticawetenschappen en CD bestaat of wordt ontwikkeld. In 2014 publiceerde de Koninklijke Vlaamse Academie van België voor Wetenschappen en Kunsten (KVAB) in samenwerking met de Jonge Academie een bijdrage rond informaticawetenschappen in het leerplichtonderwijs [25]. Dit standpunt werd uitgewerkt nadat in enkele buurlanden de desbetreffende academies hierover standpunten publiceerden [29, 18]. Onder andere in Nederland is men op aangeven van een gelijkaardig rapport [18] begonnen met het invullen van informaticawetenschappen binnen het leerplichtonderwijs. In het Vlaamse rapport formuleert de KVAB enkele aanbevelingen rond het inwerken van informaticawetenschappen in het leerplichtonderwijs. De twee hoofdaanbevelingen zijn:

> ***1.*** *Zowel in het basisonderwijs als in het secundair onderwijs dient een sterke component informaticawetenschappen opgenomen te worden in het leerplichtonderwijs. De op stapel staande onderwijshervorming biedt hiertoe een unieke kans* [25, p. 4, 45].

> ***2.*** *Om degelijk onderwijs in de informaticawetenschappen te kunnen aanbieden, dienen de lerarenopleidingen inhoudelijk aangepast te worden en aantrekkelijker gemaakt. Tegelijk dient op korte en middellange termijn sterk ingezet te worden op bijscholing van het bestaande leerkrachtenkorps* [25, p. 4, 45].

De KVAB volgt hierin de eerdere adviezen van de academies uit onze buurlanden en betrekt ze op het Vlaams onderwijs en de aanstaande onderwijshervorming. Computationeel denken is voor de Vlaamse Academie een belangrijk begrip in zowel informaticavaardigheden als informaticawetenschappen en de Academie licht CD uitvoerig toe in hun standpunt [25]. Computationeel denken aanleren is volgens de academie des te belangrijker omdat CD de universele principes van informaticawetenschappen omvat die ook overdraagbaar zijn naar andere contexten [25].

[7] In Nederland is de situatie complexer dan in het VK, maar is er een nieuwe (en gedeeltelijk vernieuwde) invulling voor het bestaande keuzevak “informatica” uitgedacht. Het vernieuwde keuzevak zal kunnen worden opgenomen door leerlingen uit de bovenbouw in het VWO en HAVO.

> *De informaticawetenschappen zijn gebaseerd op een aantal **universele principes** die 'overdraagbaar' zijn naar vele andere contexten en die daar hun nut kunnen bewijzen. De verzameling van deze principes wordt vaak gezamenlijk aangeduid met de term **computationeel denken**, en omvat, onder andere, logisch en gestructureerde redeneertechnieken, vermogen tot abstractie, voorstellen van gegevens, algoritmisch denken, opdelen van problemen in deelstappen, etc. Computationeel denken kan worden gezien als een deelaspect van het bredere begrip 'probleemoplossend denken'. De informaticawetenschappen bieden een natuurlijk, actueel en toekomstgericht kader voor het aanleren van computationeel denken* [25, p. 11-12].

Voorlopig zijn weinig van de aanbevelingen uit het standpunt doorgevoerd in het Vlaams secundair onderwijs. Of de onderwijshervorming, waarvan de invoering met een jaar uitgesteld is tot september 2019, deze aanbevelingen wel omzet in eindtermen en leerplannen valt nog af te wachten.

2.1.5 Vergelijking met probleemoplossend denken

De lezer vraagt zich misschien af of er een verschil is tussen computationeel denken en probleemoplossend denken. De twee begrippen worden in de praktijk vaak door elkaar gebruikt. Daarom is het belangrijk om te wijzen op een aantal gelijkenissen en verschillen. We beperken onze vergelijking hier tot de volgende elementen van computationeel denken: *abstractie*, *veralgemening*, *decompositie*, *patroonherkenning* en *evaluatie*. Probleemoplossend denken is, zoals CD, gebaseerd op het gebruik van een reeks technieken om problemen (leren) op te lossen. Probleemoplossend denken is een begrip dat veel eerder dan CD het licht zag. Het verscheen in een tijdperk waarin de ontwikkeling van computers nog niet aan de orde was. Mede daardoor is probleemoplossend denken ontstaan binnen het domein van de wiskunde.

In 1945 publiceert de Hongaarse wiskundige George Pólya het boek "How to Solve It" [24]. De auteur somt hierin vier basisprincipes op om het oplossen van problemen aan te leren. Hij illustreert de vier principes telkens aan de hand van één of meerdere wiskundige voorbeelden. Verschillende elementen van CD komen ook aan bod in die basisprincipes. In het eerste basisprincipe "*het probleem begrijpen*" moeten de gegevens uit de probleemstelling geanalyseerd en geïnterpreteerd worden. Hierbij wordt vaak aan *abstractie* gedaan omdat een geschikte notatie moet worden gevonden om de probleemruimte voor te stellen [24, p. 7]. Het tweede basisprincipe "*een plan bedenken*" vertoont elementen van CD zoals *veralgemening*[8], *decompositie* [24, p. 114] en *patroonherkenning* [24, p. 108]. De twee laatste principes "*het plan uitvoeren*" [24, p. 12-14] en "*de oplossing opnieuw bekijken*" [24, p. 14-20] vertonen vooral aspecten van *evaluatie*. Pólya stelt dat tijdens en na het uitvoeren van het plan er grondig moet nagekeken worden of de oplossing(smethode) correct is. Dit zal helpen om

[8] [24, p. 9]: "*Do you know a related problem?* [...] *Look at the unknown! And try to think of a familiar problem having the same or a similar problem.*".
[24, p. 108]: *Generalization.*

de oplossing beter te begrijpen en na te gaan of er nog een betere oplossing mogelijk is.

Bij de gedetailleerde bespreking van de componenten van CD (onderdeel 2.3) wordt, indien van toepassing, telkens beschreven op welke manier een kernelement anders aan bod komt bij computationeel dan bij probleemoplossend denken.

2.2 Situering

Amerikaans onderzoeker Seymour Papert beschreef als een van de eersten hoe belangrijk het was dat informatica zo snel mogelijk een vaste plaats moest krijgen binnen het klaslokaal. Dit deed hij al in "Mindstorms" [23] uit 1980, hetzelfde werk als dat waarin hij als eerste verwijst naar de term *computational thinking*. In "Mindstorms" buigt hij zich over hoe kinderen moeten leren omgaan met computers. Daarbij doelt hij niet zozeer op het leren gebruiken van computers voor bijvoorbeeld tekstverwerking, maar op het programmeren of bedienen van de machine als creatieve aangelegenheid. De computer is er als baanbrekend hulpmiddel voor het oplossen van problemen. De ideeën die hij opsomt over het gebruik van een computer zijn gelijkaardig met wat vandaag de dag computationeel denken genoemd wordt:

> *In many schools today, the phrase "computer-aided instruction" means making the computer teach the child. One might say the computer is being used to program the child. In my vision, "the child programs the computer" and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building* [23, p. 182].

Papert verwijst naar hoe leerlingen zelf computers moeten ontdekken en de digitale *tools* geen leerstof moeten voorkauwen. De computer is meer dan louter een leermiddel. In het Vlaams secundair onderwijs wordt nog altijd veel aandacht besteed aan het leren werken met tekstverwerkingspakketten zoals Microsoft Word of het maken van presentaties in bijvoorbeeld Microsoft PowerPoint of Prezi. Deze werkwijze is erg kortzichtig aangezien software en technologieën razendsnel evolueren en veranderen. Er is hierin ook weinig ruimte voor creatieve verkenning buiten de methodes die worden aangeleerd of voor het (zelf) ontdekken van nieuwe functies op de krachtige toestellen van vandaag. Er is meer aandacht nodig voor het leren doorgronden van de technologie zelf en een bredere, toekomstgerichte invulling van de les informatica is noodzakelijk. Wanneer informaticawetenschappen als onderdeel zou worden opgenomen in het leerplichtonderwijs (zie 2.1.4) dient geschikt materiaal te worden ontworpen dat leerkrachten in hun taak ondersteunt. Wij zijn van mening dat computationeel denken expliciet benoemd moeten worden in dat lesmateriaal. Hetzelfde geldt voor de vaardigheden en attitudes waarop getraind wordt. Het lesmateriaal op Co-De tracht deze denkwijze te illustreren. Op deze manier groeien de vaardigheden van CD uit tot concepten die elk individu de rest van zijn leven meedraagt.

2.3 Vijf kernelementen

Zoals vele auteurs definiëren we in dit onderzoeksproject computationeel denken aan de hand van enkele kernelementen. Deze kernelementen of -vaardigheden zijn gebaseerd op de praktijkgerichte uitwerking van *CAS* [9] (tabel 2.4 op pagina 11). Door slechts vijf componenten te selecteren, vormen die een bevattelijk geheel van begrippen waarnaar eenvoudig kan worden gerefereerd. De vijf kernelementen zijn bovendien de elementen die in tabel 2.4 het vaakst voorkomen. Er is geen rangorde tussen de verschillende elementen en de inhoud ervan overlapt minimaal.

Computationeel denken bestaat voor ons uit de volgende vijf elementen:

Figuur 2.1: De vijf kernelementen van computationeel denken voor Co-De: *abstractie*, *veralgemening*, *decompositie*, *algoritmisch denken* en *evaluatie*. De elementen worden begeleid door hun symbolen en hebben een vaste kleur.

Deze vijf kernelementen zijn vaardigheden die kunnen worden aangeleerd door training en bewustwording. Het leerplatform vermeldt ze dan ook expliciet voor zowel de leerkrachten als de leerlingen. Men kan immers slechts goed leren computationeel denken als men actief nadenkt over welke vaardigheden men op welke plaats kan gebruiken.

Om deze kernbegrippen herkenbaar te maken, hebben we symbolen ontwikkeld die gekoppeld zijn aan elementen van computationeel denken (zie figuur 2.1). Elk symbool tracht het begrip te karakteriseren aan de hand van een duidelijk beeld en een eigen kleur. Deze symbolen worden intensief gebruikt op het leerplatform en zullen ook in deze tekst regelmatig terugkeren.

De vijf vaardigheden worden hieronder één voor één besproken. Elke vaardigheid wordt eerst gedefinieerd en de definitie wordt verduidelijkt aan de hand van enkele voorbeelden. Indien er verschillen zijn ten opzichte van probleemoplossend denken (zie 2.1.5) worden deze ook toegelicht. Tot slot wordt ook aangegeven welk symbool verbonden is aan de vaardigheid en waarom.

2.3.1 Abstractie

Abstractie is het achterwege laten van details die niet van belang zijn om tot een oplossing te komen. Minder details kunnen immers helpen om een beter zicht te krijgen op het probleem.

Deze vaardigheid is belangrijk om (ogenschijnlijk) complexe problemen op te lossen. Via *abstractie* kunnen overbodige elementen achterwege worden gelaten. Dit geeft op zijn beurt aanleiding tot het wegwerken van variabelen en ingewikkelde regels te vereenvoudigen. Een ander aspect dat van belang is, is het uitwerken of kiezen van een goede datarepresentatie.

2.3.1.1 Een voorbeeld

Bij het maken van een realistische tekening op papier heeft men enkel informatie nodig over vormen, kleuren, afmetingen en dergelijke. Elementen zoals geuren of geluiden zijn hier overbodig. Ook de objecten die buiten het papier vallen zijn (in de meeste gevallen) niet van belang.

Men kan dus sneller en eenvoudiger een tekening maken als men *abstractie* maakt van de scène en enkel de elementen in rekening neemt die daadwerkelijk van belang zijn voor de tekening.

2.3.1.2 Verschil met probleemoplossend denken

Het niveau waarop *abstractie* gemaakt moet worden, ligt bij computationeel denken vaak hoger dan bij probleemoplossend denken. Bij CD moet men namelijk een probleemstelling in de echte wereld abstraheren terwijl bij wiskundig en algemeen probleemoplossend denken vaak reeds een deel van de *abstractie* voorzien is. Een wiskundig probleem wordt immers meestal al gepresenteerd in een abstracte vorm met symbolen en variabelen. Bovendien moet men bij probleemoplossend denken minder ver gaan in de *abstractie*, aangezien het doorgaans niet nodig is om datarepresentaties en algoritmes uit te denken die een computer begrijpt. Ten slotte komt *abstractie* vaak in een andere grootteorde naar voren bij CD. Een consistent systeem uitwerken om een dienstregeling van het treinverkeer in België op te stellen, vraagt *abstractie* op grotere schaal dan *abstractie* bij het opstellen van een formule om te bepalen hoe groot de kans is dat je voor twee gesloten slagbomen zal staan (gegeven een bepaalde kansverdeling van de twee seinen).

2.3.1.3 Symbool

Een rood, van nature vrij abstract symbool stelt het kernelement *abstractie* voor. Het symbool bestaat uit enkele cirkels met verschillende diameters. Toch zijn het allemaal cirkels. Het zijn dus, als men *abstractie* maakt van de verschillende diameters, allemaal dezelfde objecten. Bovendien vormen ze samen ook een cirkel.

2.3.2 Veralgemening

Veralgemening bestaat uit twee soorten. Men kan het probleem en/of de oplossingsmethode veralgemenen. Om het onderscheid te illustreren volgt hieronder een voorbeeld van beide soorten (zie 2.3.2.1).

Veralgemening heeft als doel om oplossingsmethodes te bekomen die breder toepasbaar zijn dan enkel voor het specifieke probleem waarvoor ze werden opgesteld. Ook een oplossingsmethode leren toepassen in (soortgelijke) probleemsituaties behoort tot de vaardigheid *veralgemening*. Dit is cruciaal in een leerproces en dus ook bij het leren computationeel denken.

2.3.2.1 Voorbeelden

Een voorbeeld van *veralgemening* van het probleem:
"Één cirkel tekenen" veralgemenen tot
"verschillende cirkels tekenen met verschillende diameters".

Het eerste zou men kunnen doen door een rond voorwerp te zoeken met de grootte van de cirkel die men wil tekenen. Door met een pen rond het voorwerp te tekenen kan men eenvoudig de cirkel overnemen op papier. Het is nu mogelijk om verschillende cirkels met dezelfde straal te tekenen. Om cirkels met verschillende stralen te tekenen zou men verschillende objecten moeten zoeken en bijhouden.

Een alternatief is om via een touwtje aan de pen een soort passer te maken. Op deze manier kan men nog steeds eenvoudig cirkels met dezelfde straal tekenen, maar is het veel gemakkelijker de straal te laten variëren. Deze aanpak is dus algemener dan de eerste. Bij het zoeken naar een oplossingsmethode zullen we deze methode verkiezen boven de eerste op het vlak van algemeenheid.

Een voorbeeld van *veralgemening* van de oplossingsmethode:
"Een oplossingsmethode voor lineaire stelsels met coëfficiënten in $\mathbb{N}$" veralgemenen tot "Een oplossingsmethode voor lineaire stelsels met coëfficiënten in $\mathbb{Q}$".

Aangezien $\mathbb{Q} \supset \mathbb{N}$ lost elke methode die lineaire stelsels met coëfficiënten in $\mathbb{Q}$ oplost ook lineaire stelsels met coëfficiënten in $\mathbb{N}$ op. De oplossingsmethode voor $\mathbb{Q}$ is bijgevolg algemener en dus, op het gebied van algemeenheid, beter.

2.3.2.2 Symbool

Een geel symbool met pijlen stelt het element *Veralgemening* voor. De pijlen wijzen naar de buitenkant omdat men bij *veralgemening* tracht om een oplossingsmethode te construeren die breed toepasbaar is. Door allerlei soorten variaties en soortgelijke problemen te beschouwen, leert men de oplossingsmethodes ook echt toe te passen.

2.3.3 Decompositie

Decompositie kan worden omschreven als: het opdelen van een (complex) probleem in kleinere, meestal eenvoudigere delen die makkelijker op te lossen zijn. Nadien worden de oplossingen van de deelproblemen gecombineerd tot een gezamenlijke oplossing. Het gaat bij *decompositie* niet zo zeer over het opdelen van de oplossingsmethode in verschillende delen of stappen, maar om het opdelen van het probleem in deelproblemen die elk op zich moeten worden opgelost.

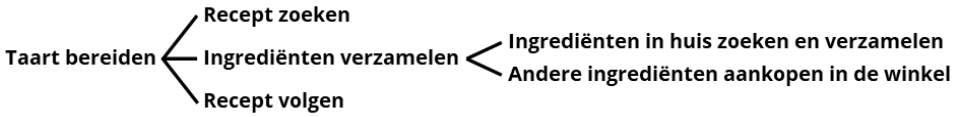
Men kan aan *decompositie* doen op twee manieren. Enerzijds kan men een probleem opdelen in instantiaties van hetzelfde probleem. Anderzijds is het mogelijk om een probleem te decomposeren in deelproblemen van verschillende aard. Hieronder volgt een voorbeeld van elke soort.

2.3.3.1 Voorbeelden

Een voorbeeld van *decompositie* in instanties van hetzelfde probleem:
Een moeilijke vermenigvuldiging (bv. $357 \cdot 223$) kan men oplossen door ze op te delen in kleinere en eenvoudigere vermenigvuldigingen en de uitkomsten daarvan op te tellen. Zo bekomt men:

$$\begin{array}{lcccccc} 357 \cdot 223 & = & (300 \cdot 200) & + & (50 \cdot 200) & + & (7 \cdot 200) \\ & + & (300 \cdot 20) & + & (50 \cdot 20) & + & (7 \cdot 20) \\ & + & (300 \cdot 3) & + & (50 \cdot 3) & + & (7 \cdot 3) \end{array}$$

Een voorbeeld van *decompositie* in deelproblemen van verschillende aard:
Wanneer men een meringuetaart wilt maken moet men verschillende problemen oplossen. Deze problemen hebben weinig met elkaar te maken en kunnen los van elkaar worden opgelost. De oplossingsmethode is ook erg verschillend voor elk deelprobleem. De afbeelding hieronder geeft enkele van deze deelproblemen weer:



Figuur 2.2: Een *decompositie* in deelproblemen van verschillende aard bij het probleem "een meringuetaart maken".

2.3.3.2 Verschil met probleemoplossend denken

De onderliggende betekenis van *decompositie* is dezelfde bij probleemoplossend en computationeel denken, namelijk het opsplitsen in deelproblemen. De schaal waarop decompositie wordt toegepast is echter vaak verschillend. Probleemoplossend denken blijft meestal beperkt tot het domein van de wiskunde, zoals duidelijk is in het eerste voorbeeld van 2.3.3.1 (vermenigvuldiging). Bij computationeel denken beperken we ons niet tot een bepaald domein en willen we realistische problemen oplossen. In het tweede voorbeeld (het maken van een taart) is dat het geval. De manier waarop

de deelproblemen moeten worden aangepakt, zijn daardoor afhankelijk geworden van de context: “Welke infrastructuur is er in de keuken aanwezig?” of “Worden de aanwezige ingrediënten nog voor andere zaken gebruikt?”. Nog complexer wordt het wanneer het probleem nog omvangrijker is zoals het organiseren van de *dataflow* in een hospitaal: “Hoe zorgt men ervoor dat elke dienst of persoon de juiste gegevens kan raadplegen?”, “Hoe wordt de privacy van patiëntengegevens gewaarborgd?”...

2.3.3.3 Symbool

Het symbool voor *decompositie* is groen en bestaat uit vier puzzelstukken. Samen vormen de kleine puzzelstukken het volledige symbool, naar analogie met het originele probleem dat wordt opgedeeld in deelproblemen. De oplossingen van die deelproblemen vormen daarna, als puzzelstukken, de oplossing van het geheel.

2.3.4 Algoritmisch denken

Algoritmisch denken omvat een stappenplan bedenken waarmee een probleem steevast kan worden opgelost. Dit stappenplan is een opeenvolging van instructies die stap-per-stap moeten worden uitgevoerd en die steeds van een probleemstelling tot een oplossing zullen leiden als die bestaat. De instructies moeten ondubbelzinnig aangeven wat er in welke volgorde moet gebeuren.

Algoritmisch denken als vaardigheid binnen CD is niet beperkt tot een algoritme implementeren op een computer. Algoritmes programmeren valt hier wel onder, maar men kan ook leren *algoritmisch denken* zonder computer.

2.3.4.1 Een voorbeeld

Een recept voor chocomousse beschrijft stap voor stap wat men moet doen om van een reeks ingrediënten een lekkere chocomousse te maken. Die opeenvolging van stappen in de keuken vormt een algoritme.

Indien het recept niet duidelijk is, kan het proces fout aflopen. Ook wanneer men de stappen niet in de juiste volgorde doorloopt, kan de chocomousse mislukken. Als men echter alle stappen nauwgezet en in de juiste volgorde doorloopt, bekomt men elke keer opnieuw een lekkere chocomousse.

2.3.4.2 Verschil met probleemoplossend denken

Algoritmisch denken vormt het grootste verschil tussen computationeel en probleemoplossend denken. Het algoritmisch aspect is veel uitgebreider en explicieter aanwezig binnen CD. Ook voor problemen die niet over informatica gaan, maar gelinkt zijn aan andere domeinen, vormt een algoritme vaak de essentie van een oplossingsmethode. Dit is bijvoorbeeld het geval wanneer de structuur van het menselijk DNA wordt geanalyseerd. Dankzij algoritmisch denken is het mogelijk om een aantal concrete

eisen te stellen aan een uitgewerkte oplossingsmethode, zoals dat een algoritme steeds hetzelfde resultaat moet opleveren bij dezelfde input (reproduceerbaarheid). Algoritmisch denken binnen CD kan eveneens helpen om de oplossingsmethode op een formele manier voor te stellen. Bij probleemoplossend denken wordt hier niet altijd aandacht aan besteed. Een bewijs is daar geregeld het eindpunt maar dat vormt daarom nog geen algoritme.

2.3.4.3 Symbool

Een blauw symbool met enkele horizontale balkjes en een verticale pijl beeldt het kernelement *Algoritmisch denken* af. De balkjes illustreren een reeks instructies of stappen uit een algoritme. De pijl geeft aan dat men deze stappen in een welbepaalde volgorde moet uitvoeren.

2.3.5 Evaluatie

Onder *evaluatie* valt het nakijken van de oplossing(smethode) aan de hand van bepaalde maatstaven. Evalueren is niet iets dat men pas achteraf moet doen. Men denkt hier best ook al over na tijdens de zoektocht naar een oplossing en het ontwerp van een oplossingsmethode.

De maatstaven die worden gebruikt kunnen variëren per probleem of situatie. In informaticawetenschappen worden vaak tijds- en geheugencomplexiteit gebruikt, maar dit zijn zeker niet de enige maatstaven die van belang zijn. Ook de correctheid en optimaliteit zijn belangrijk en men stelt zich best vragen als “Zijn alle stappen in de methode nodig?” of “Vindt de methode snel genoeg een oplossing?”. Welke hulpmiddelen er nodig zijn, kan ook een rol spelen. Hulpmiddelen bestaan in allerlei gedaantes: oplossingsmethodes uit eerdere opdrachten, externe tools of bibliotheken, aantal personen, aantal man-uren, plaatsgebruik, tekenmateriaal...

2.3.5.1 Een voorbeeld

Stel dat men een zakje M&M’s wil splitsen per kleur via de volgende methode:

1. Haal alle rode M&M’s uit de zak en leg ze apart.
2. Haal alle blauwe M&M’s uit de zak en leg ze apart.
3. Haal alle gele M&M’s uit de zak en leg ze apart.
4. Haal alle groene M&M’s uit de zak en leg ze apart.
5. Haal alle oranje M&M’s uit de zak en leg ze apart.
6. Haal alle bruine M&M’s uit de zak en leg ze apart.

Dit stappenplan zal zeker werken, maar eigenlijk doet men een stap te veel. Men kan de laatste stap achterwege laten aangezien er na stap 5 enkel nog bruine M&M’s overblijven en het probleem dus al is opgelost.

Een alternatieve oplossingsmethode is om zes mensen elk een kleur te laten verzamelen. Deze methode is waarschijnlijk sneller in tijd, maar vergt meer hulpmiddelen (mensen) en meer afspraken onderling (een protocol) om dit probleemloos te laten verlopen.

2.3.5.2 Verschil met probleemoplossend denken

Zowel bij computationeel als bij probleemoplossend denken doet men aan *evaluatie*, bijvoorbeeld om de correctheid van een oplossing(smethode) aan te tonen. Bij probleemoplossend denken gebeurt dit meestal aan de hand van een bewijs. Ten eerste is dit niet altijd mogelijk voor problemen uit het dagelijkse leven die we met CD trachten op te lossen. Je zal bijvoorbeeld nooit kunnen bewijzen dat als je naar de winkel gaat met een boodschappenlijst, je deze producten 100% zeker zal kunnen aankopen en meenemen naar huis. Ten tweede is enkel een bewijs van correctheid niet voldoende bij computationeel denken. Vaak speelt ook efficiëntie een rol of zijn er beperkingen op het gebruik van hulpmiddelen (tijd, mensen, plaatsgebruik...). Een bepaalde oplossingsmethode kan dus “beter” zijn dan een andere, ondanks dat beide correct zijn. Bij problemen die door de computer worden opgelost, is er bovendien beperkte rekenkracht en geheugen voorhanden.

2.3.5.3 Symbool

Een vergrootglas op een paarse achtergrond vormt het symbool voor *evaluatie*. Het vergrootglas illustreert dat men de oplossing en de oplossingsmethode onder de loep moet nemen. Er wordt gezocht naar goede en minder goede aspecten van de oplossing(smethode) en alternatieven worden overwogen.

2.4 Overige aspecten

Het is belangrijk om te vermelden dat de vijf besproken elementen van computationeel denken niet exclusief zijn. Er bestaat een zekere overlap tussen de verschillende elementen. Zo zijn er bijvoorbeeld bepaalde gelijkenissen tussen *decompositie* en *algoritmisch denken*. Bij beide zal men immers een stappenplan moeten maken om te bepalen in welke volgorde men te werk gaat.

Zoals duidelijk blijkt uit het debat (zie 2.1.2) bestaan er verschillende gangbare interpretaties van computationeel denken. Er zijn dus wellicht aspecten van CD die in onze interpretatie onderbelicht worden. Dit betekent echter niet dat er geen ruimte is op Co-De om deze te behandelen, maar deze zullen dan slechts beperkt gekoppeld kunnen worden aan de vijf elementen van CD.

Wij zijn er desalniettemin van overtuigd dat de vijf elementen voor Co-De, gebaseerd op de interpretatie van *Computing at School* (*CAS*) [9], een goede basis bieden om van te vertrekken. Deze basisvaardigheden zijn immers brede begrippen waar heel wat andere vaardigheden en attitudes uit voortvloeien. Nogal wat elementen die aan bod komen in andere interpretaties uit het debat (2.1.2) vinden volgens *CAS* [9] een plaats onder één of meerdere van de vijf vaardigheden voor Co-De.

Een goede datarepresentatie vinden en aan data-analyse doen, komen bijvoorbeeld voort uit een combinatie van *abstractie*, *veralgemening* en *decompositie*. Via *abstractie* kan men een probleem en de data analyseren, structureren en abstraheren. Door *decompositie* kan men die doordacht opdelen. Een goede datarepresentatie is ook algemeen inzetbaar en herbruikbaar in andere situaties, daar komt *veralgemening* bij kijken.

Transfer, het overhevelen van oplossingsmethodes naar andere problemen, situaties en domeinen, sluit nauw aan bij *veralgemening*. In de literatuur is er vaak veel aandacht voor transfer aangezien men pas echt iets leert wanneer men ook oog heeft voor toepassingen in nieuwe situaties [9, p. 14]. Er zijn echter weinig auteurs die transfer bestempelen als een kernelement van CD aangezien andere elementen reeds inzetten op transfermogelijkheden. Daarom wordt transfer ook niet expliciet vernoemd in het debat (2.1.2). Naast *veralgemening* kunnen ook *abstractie* en *evaluatie* een rol spelen bij transfer. Op een abstracter niveau kan men immers sneller linken leggen tussen verschillende situaties en problemen. Men houdt best ook rekening met de transfermogelijkheden bij de *evaluatie*.

Patroonherkenning wordt vaak afzonderlijk vernoemd als kernelement, maar zit voor ons vervat in *veralgemening*. Door de patronen in verschillende problemen af te leiden, kan een oplossingsmethode voor één van die problemen misschien hergebruikt worden. Patroonherkenning zal zeker aan bod komen op Co-De, maar als onderdeel van *veralgemening*. Op lager niveau speelt ook *abstractie* een rol bij patroonherkenning.

Ook logisch redeneren, debuggen, automatiseren en tal van andere vaardigheden kan men op deze manier verbinden aan de vijf kernvaardigheden van CD voor Co-De. Door het aantal vaardigheden beperkt te houden, verhoogt de herkenbaarheid en wordt het begrip CD bevattelijk. Andere zinvolle vaardigheden (bv. logisch redeneren) en attitudes (bv. aandacht voor debuggen) worden dus wel aangeleerd, maar niet bestempeld als een kernelement van CD.

2.5 Besluit

Uit de uitgebreide literatuurstudie over computationeel denken (CD) blijkt dat er heel wat interpretaties van het begrip bestaan. Het was Seymour Papert die de term voor het eerst vermeldde in 1980. Bijna 30 jaar later, in 2006, formuleerde Jeanette Wing haar definitie. Wings uiteenzetting gaf meteen aanleiding tot een debat over de concrete invulling van CD. Verschillende organisaties verwoordden hun eigen invulling van het begrip CD, doorgaans aan de hand van enkele kernelementen. Het debat is nog steeds levendig en er is nog geen algemene consensus over een definitie van CD. Dat staat echter niet in de weg dat er in de praktijk heel wat rond CD gewerkt wordt. De initiatieven van *cs4fn* en *CAS* in het VK, de wereldwijde *Bebras Computing Challenge* en de recente uitgave "Zo denkt een computer" zijn daar voorbeelden van. Wanneer men de interpretaties van CD bekijkt die deze voorbeelden hanteren, blijkt dat er in praktijk wel enige consensus bestaat over de invulling van CD. Tabel 2.4 (p. 11) vat deze bevindingen samen.

Vertrekkende van deze consensus werden vijf kernelementen van CD geïdentificeerd die een belangrijke rol spelen voor Co-De. Deze vijf elementen of vaardigheden zijn: *abstractie*, *veralgemening*, *decompositie*, *algoritmisch denken* en *evaluatie*. De kernelementen werden één voor één uitgebreid toegelicht en gekaderd door middel van enkele voorbeelden. Daarbij was ook aandacht voor de verschillen ten opzichte van andere denkwijzen zoals probleemoplossend en wiskundig denken. Daarenboven werden herkenbare symbolen gecreëerd voor de verschillende componenten.

Ook over het belang van CD is er eensgezindheid. Computationeel denken zou een vaardigheid moeten zijn die iedereen beheerst naast lezen en schrijven. Hierin speelt het onderwijs een belangrijke rol. Dat computationeel denken een plaats moet krijgen binnen het onderwijs wordt ondersteund door de rapporten van academies in Vlaanderen en in onze buurlanden. Op sommige plaatsen zet men reeds sterk in op CD binnen het informatica-onderwijs. Dat is in het secundair onderwijs in Vlaanderen (nog) niet het geval. Of de op stapel staande onderwijshervorming daar verandering in zal brengen, valt nog af te wachten.

Hoofdstuk 3

Co-De Leerplatform

3.1 Inleiding

Dit hoofdstuk zoomt in op de ontwikkeling en het gebruik van het leerplatform Co-De. Co-De [koː-dɛ] staat voor de eerste lettergreep uit beide woorden van "computationeel denken". Het leerplatform is sinds februari 2018 toegankelijk op `https://code.experiments.cs.kuleuven.be` (in het vervolg afgekort tot `$code`). Elke gebruiker moet zich registreren op Co-De vooraleer hij de inhoud kan bekijken. De server van Co-De bevindt zich in het departement computerwetenschappen van de KU Leuven.

De inhoud van dit hoofdstuk is als volgt opgebouwd. Ten eerste volgt een beknopt overzicht van de belangrijkste ontwikkelingen in verband met *e-learning* en online leerplatformen vanaf de jaren 1980. Ten tweede komt de selectie van het leerplatform aan bod. Ten derde volgen, meer in detail, de belangrijkste keuzes voor de technische opbouw van Co-De. Ten vierde wordt uitgelegd hoe lessen op Co-De worden geconstrueerd en welke functionele aanpassingen daarvoor nodig waren. Die bespreking beperkt zich tot de lesstructuur, het lesverloop, de soorten lesactiviteiten en de gebruikersrollen. De inhoudelijke opbouw van specifieke lessen komt aan bod in hoofdstuk 4. Ten vijfde worden de mogelijkheden als gebruiker van Co-De besproken. Hierbij wordt herhaaldelijk verwezen naar de gebruikershandleiding die online beschikbaar is. Ten zesde is er nog een deel gewijd aan de testen waaraan het leerplatform werd onderworpen. Tot slot volgt een besluit.

3.2 Literatuurstudie

Begrippen zoals multimediaal leren, *e-learning* en online leren zijn niet meer nieuw. Nog vooraleer de term computationeel denken ontstond, werd er over deze concepten nagedacht. Al vrij snel volgden er, in de jaren 1990, uit deze discussies cognitieve paradigma's voor het ontwerpen van digitale leertools [20]. Tegelijkertijd ontstonden de eerste online leerplatformen en tools. Rond het jaar 2000 verschijnen enkele van de vandaag nog populaire leerplatformen, zoals Blackboard Learn [4], Moodle [22] en Smartschool [28].

Pedagogisch gezien mogen de aspecten van digitaal leren dan wel grondig bestudeerd zijn [19], op het vlak van software-ontwerp en -architectuur zijn er tot op de dag van vandaag weinig pasklare antwoorden. Als belangrijkste bron geldt nog steeds de documentatie van bestaande platformen. Blackboard Learn en Moodle, op internationaal vlak de twee meest bekende leerplatformen op dit moment, hebben beide een uitgebreid online discussieforum[1] en een goed uitgewerkte handleiding[2]. Voor Moodle is er ook een site gewijd aan nuttige informatie voor ontwikkelaars[3]. In België heeft ook Smartschool[4] een grote bekendheid. De documentatie voor Smartschool is echter enkel beschikbaar voor deelnemende scholen.

3.3 Selectie van het platform

De consequentie van de beperkte literatuur over het bouwen van een leerplatform is dat de juiste technologie kiezen geen eenvoudige kwestie is. Vooral op functioneel vlak zijn er heel wat eisen waarmee rekening moet worden gehouden voor Co-De. De belangrijkste onderzochte afweging was die tussen een volledig nieuw platform ontwikkelen of vertrekken van een bestaand platform en dit verder uitbouwen.

3.3.1 Vergelijking eigen applicatie en bestaand platform

De keuze voor één van beide opties ligt niet voor de hand. Elke optie gaat uiteraard gepaard met een aantal interessante voordelen en niet te verwaarlozen hindernissen. Om verschillende redenen werd uiteindelijk geopteerd om het *open source* leerplatform Moodle te gebruiken. We overlopen de argumenten die ons overtuigden om voor Moodle te kiezen.

Tabel 3.1 vat deze bevindingen samen. Ze toont een overzicht van de eigenschappen die voor Co-De van belang zijn. Hiervan voldoet Moodle zeer duidelijk aan het merendeel van de vereiste eigenschappen en is het bovendien gratis beschikbaar.

3.3.1.1 Eigen applicatie

Een eigen architectuur uitwerken om Co-De te creëren leek aanvankelijk de meest aantrekkelijke keuze. Dankzij een particuliere architectuur hoeven er weinig compromissen gemaakt te worden qua eigenschappen van Co-De. Het is mogelijk om precies te bepalen welke manieren van interageren noodzakelijk zijn op het platform: welke volgorde de lesonderdelen kunnen hebben en hoe je de beschikbaarheid ervan kan afstellen, welke vormen van lesinhoud ondersteund zijn (animaties, *applets*, tekst, afbeeldingen of multimedia) etc.

[1] Blackboard Learn: https://community.blackboard.com/.
Moodle: https://moodle.org/course/index.php/.
[2] Blackboard Learn: https://help.blackboard.com/.
Moodle: https://docs.moodle.org/34/en/Main_page/.
[3] Ontwikkelaarssite: https://docs.moodle.org/dev/Main_Page/.
[4] Smartschool: http://www.smartschool.be/.

	Bestaande platformen			Python web-frameworks	
	Blackboard Learn	Moodle	Smartschool	Django	Flask
Online toegang via een browser of app	Ja	Ja	Ja	Ja	Ja
Basisfunctionaliteit wijzigen mogelijk	Ja uitsluitend bij eigen hosting	Ja	Nee	Ja	Ja
Extra functionaliteit toevoegen	Ja via API	Ja via eigen plugins	Ja via API	Ja	Ja
Autorisatie ingebouwd (login)	Ja	Ja	Ja	Ja	*
Lessen exporteren en importeren (ook tussen verschillende installaties)	Ja	Ja	Ja	*	*
Leerpaden configureren	Per les	Per les	Per individu	*	*
Conditioneel verloop van de les past zich aan volgens het leerpad	Nee	Nee	Nee	*	*
Onderscheid in gebruikersrollen en in gebruikersrechten (leerling, leerkracht...)	Ja	Ja	Ja	*	*
Zelfregistratie op leerplatform	Ja	Ja	Nee	*	*
Zelf aanmelden op lessen	Ja	Ja	Nee	*	*
Zelf hosten mogelijk	Ja	Ja	Nee	Ja	Ja
Kostprijs	Per gebruiker	Gratis	Per gebruiker	Gratis	Gratis

* Deze eigenschap moet nog zelf geïmplementeerd worden in het Python web-framework.

Tabel 3.1: Het overzicht van de eigenschappen per leerplatform of web-framework.

De keerzijde van die flexibiliteit is dat er voldoende tijd gealloceerd moet zijn om de ideeën tot uitvoering te brengen. Een nieuwe applicatie opbouwen impliceert dat een bepaalde webtechnologie of framework gekozen moet worden. In de eerste plaats komen Python-webframeworks in aanmerking, omdat geïnteresseerden uit het onderwijs met die taal wellicht sneller kunnen meebouwen aan Co-De. Python groeit namelijk steeds meer uit tot de gangbare programmeertaal binnen het onderwijs [30]. Twee van zulke frameworks, Django[5] en Flask[6], werden onder de loep genomen om na te gaan of ze een goede basis zouden zijn voor Co-De. Veel aspecten die zorgen voor een goede configuratie van een leerplatform moeten nog zelf geïmplementeerd worden bij Django en Flask (asterisken in tabel 3.1): rechten en rollen van de gebruikers, bijhouden van gegevens over gebruikers, beveiliging van het platform, embedden van bepaalde inhoud... Sommige elementaire zaken, bijvoorbeeld de autorisatie van gebruikers, zijn wel al ontwikkeld voor Django, maar nog niet voor Flask (tabel 3.1). Het ontwikkelen van die algemene aspecten van het leerplatform kost veel tijd en draagt verder niet bij aan de inhoud van het platform.

3.3.1.2 Bestaand platform

Enerzijds levert werken met een bestaand platform een oplossing voor sommige beperkingen die hierboven werden aangehaald. Bestaande platformen (tabel 3.1) leveren al heel wat basisfunctionaliteiten om de rechten van gebruikers te configureren, de inhoud te beheren, de configuratie in te stellen voor een betere beveiliging, vormgeving aan te passen...

Anderzijds zullen de voorziene functies op het platform bepalend zijn voor het eindresultaat. De manier van werken met een gevestigd platform ligt a priori vast. Daarom zijn er een aantal bijkomende vereisten om met een bestaand platform verder te kunnen werken. Vooreerst is het van belang dat de nodige aanpassingen kunnen worden gedaan, al dan niet via de broncode van het platform. Blackboard, Moodle en Smartschool voldoen namelijk alle drie niet volledig aan de lesstructuur waarmee we wensen te werken. Zo worden de sequentiële condities van een leerpad niet automatisch mee aangepast als dat leerpad wordt gewijzigd (tabel 3.1, zie verder ook relatieve voltooiing in 3.5.1.2). Commerciële leeromgevingen zoals Blackboard Learn[7] en Smartschool[8] werden uiteindelijk verworpen omdat ze niet vrij beschikbaar zijn en geen wijzigingen in de broncode toelaten. Verder is er nood aan voldoende documentatie, zodat ook leerkrachten of andere vrijwilligers (op termijn) kunnen bijdragen aan Co-De.

[5] https://www.djangoproject.com/.
[6] http://flask.pocoo.org/.
[7] Blackboard Learn: http://nl.blackboard.com/.
[8] Smartschool: http://www.smartschool.be/.

3.3.2 Moodle

Moodle[9] is een *open source* leeromgeving die actief is sinds 2001. Ze volgde uit een onderzoek rond interactief leren dat werd gestart door de Australische onderzoeker en ontwikkelaar Martin Dougiamas [22]. Moodle is online beschikbaar onder de *GNU General Public License*. Naast de gewone versie die voor de gebruiker via de browser toegankelijk is, is er ook een mobiele applicatie (Moodle Mobile voor Android, iOS) en een applicatie voor desktops en tablets (Moodle Desktop). Moodle groeide uit tot een veelgebruikt platform en telt vandaag de dag wereldwijd meer dan 100 000 geregistreerde websites en meer dan 100 taalpakketten (waaronder Nederlands) [21]. Het is dus niet verwonderlijk dat Moodle in heel wat (secundaire) scholen al het standaard digitaal platform is. Dit geeft leerkrachten en ontwikkelaars de mogelijkheid om de inhoud van Co-De ook in hun Moodle-versie in te laden (zie 3.6.2.1).

De tekst verwijst verder regelmatig naar onderwerpen uit de documentatie van Moodle om de lezer voldoende context te bieden. De specifieke info die te vinden is op de vermelde webpagina's wordt daarom in de tekst zelf niet meer uitdrukkelijk overgenomen.

3.4 Technische aspecten

In dit deel worden de belangrijkste technische keuzes omtrent het opzetten van Co-De gedocumenteerd en verantwoord. Daarbij ligt de focus op een besturingssysteem-onafhankelijke installatie aan de hand van Docker-containers en het periodisch onderhoud dat ons systeem vereist.

Voor een uitgebreidere toelichting is er een technische appendix (appendix A). Daarin staat meer gedetailleerde informatie over elk van de volgende onderwerpen. Ook de belangrijkste commando's en instellingen die nodig zijn om Co-De op te zetten en te onderhouden, zijn daarin te vinden. De appendix dient voornamelijk als naslagdocument voor Co-De, maar kan evenzeer dienen als handleiding om het platform te repliceren op een andere server, zonder dat grote achtergrondkennis vereist is.

3.4.1 Deployment

We lichten kort de opbouw van onze architectuur voor Co-De toe. Onze architectuur is grotendeels afhankelijk van de mogelijkheden van de server waarmee we werken. Voor een gedetailleerdere beschrijving van de huidige opstelling kan de lezer zich wenden tot de technische appendix A.

[9] https://moodle.org/.

3.4.1.1 Machine bij departement computerwetenschappen

Het departement computerwetenschappen van de KU Leuven stelde een machine ter beschikking voor Co-De. De machine is publiek toegankelijk op `https://code.experiments.cs.kuleuven.be` (`$code`). Een beheerder van Co-De heeft ook toegang door middel van een `SSH`-tunnel. Zo kunnen de applicatie, back-ups en (multimedia)bestanden in de *home directory* van Co-De beheerd worden. Noodzakelijke bestanden worden op de server geplaatst dankzij het veilige `SFTP`-protocol. De beheerder heeft geen nood aan *root*-toestemming op de machine. Dankzij de Docker-installatie van Co-De kunnen de permissies van de mappen aangepast worden met behulp van containers.

3.4.1.2 Docker-installatie van Moodle

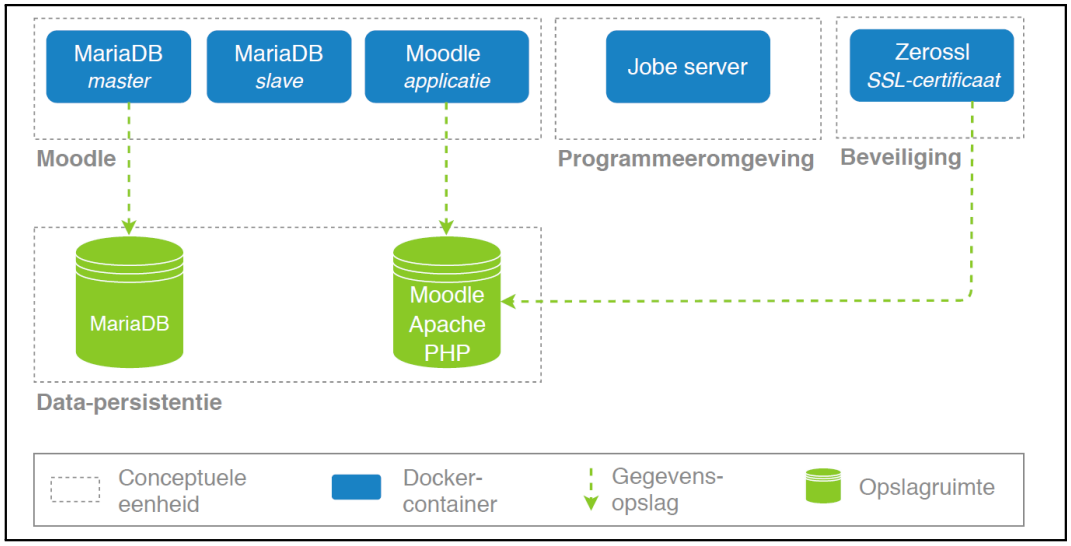
Docker is een *open source* applicatie die faciliteert om programma's en applicaties in platformonafhankelijke, geïsoleerde containers te verpakken[10]. Docker zorgt ervoor dat er geen aparte *dependencies* geïnstalleerd moeten worden. Voor Moodle betekent dit dat PHP, Apache en een SQL-databasesysteem niet vooraf geïnstalleerd moeten zijn op de server. Onafhankelijk van wat voor server men gebruikt, zorgt Docker ervoor dat een applicatie opzetten telkens via de dezelfde stappen gebeurt.

Aangezien Co-De binnen Docker wordt opgestart, zijn er slechts beperkte gebruikersrechten nodig op de machine. Enkel Docker en gebruikelijke terminalcommando's moeten voor de beheerders van Co-De toegankelijk zijn. De rest van het onderhoud van de machine is toevertrouwd aan de computerbeheerders van het departement computerwetenschappen.

Om een Docker-container op te starten, is een bijhorende *image* van het gewenste programma noodzakelijk. Co-De gebruikt de Bitnami Moodle-*image*, die beschikbaar is op `https://bitnami.com/stack/moodle`. Die Moodle-*image* maakt onderliggend gebruik van een MariaDB-*image* als databasesysteem[11].

Schijfruimte om de data van Moodle en de bijhorende database te persisteren, moet zelf nog worden gealloceerd en geconfigureerd. Figuur 3.1 toont hoe Co-De wordt gedeployed dankzij Docker. De conceptuele Moodle-entiteit (links) bestaat uit drie containers, namelijk de MariaDB-*master-slave*-containers en de Moodle-applicatie zelf (met Apache en PHP). Hiervan worden zowel de *master*-database als de Moodle-applicatie gepersisteerd. De commando's waarmee de containers en volumes opgestart worden, zijn terug te vinden in appendix A (zie A.1).

[10] `https://www.docker.com/what-docker/`.
[11] `https://hub.docker.com/r/bitnami/mariadb/`.



Figuur 3.1: Het deployment van Co-De aan de hand van vijf Docker-containers. De data uit de database en de Moodle-applicatie worden daarnaast samen met het SSL-certificaat gepersisteerd op twee schijfruimtes.

3.4.1.3 Updates

Updates van Moodle

Co-De werkt op een stabiele versie van Moodle, namelijk 3.4.1[12]. Updates kunnen op termijn manueel geïnstalleerd worden in de Moodle-container, zoals in de Bitnami Moodle-documentatie beschreven staat[13]. Hierbij is enige voorzichtigheid geboden om geen data te verliezen. Dit wordt best eerst op een afzonderlijke installatie getest.

Updates van plugins

Updates van plugins kunnen rechtstreeks binnen Moodle geïnstalleerd worden door een beheerder van Co-De[14]. Beheerders krijgen automatisch een melding wanneer een nieuwe versie van een geïnstalleerde plugin beschikbaar is.

Voor de voortgangsbalk, die gerealiseerd is met de plugin "Completion Progress"[15] (zie 3.5.1.2), is het installeren van de update zelf niet voldoende. Er zijn namelijk wijzigingen aangebracht in de broncode van de plugin. Die wijzigingen moeten ook telkens in nieuwe versies van de plugin manueel aangebracht worden. De wijzigingen worden beschreven in de technische appendix en kunnen van daar overgenomen worden om in een geïnstalleerde update aan te brengen.

[12] https://docs.moodle.org/dev/Moodle_3.4.1_release_notes.
[13] Moodle-update installeren:
https://github.com/bitnami/bitnami-docker-moodle#upgrade-this-application.
[14] https://docs.moodle.org/34/en/Installing_plugins#Installing_a_plugin.
[15] https://moodle.org/plugins/block_completion_progress.

3.4.1.4 Onderhoudstaken

Moodle laat alle geplande onderhoudstaken uitvoeren aan de hand van een PHP-script (`cron.php`[16]). Dit script moet na het opstarten van de Docker-containers nog zelf geconfigureerd worden. Dit script wordt op Co-De elke tien minuten uitgevoerd. Dit is minder frequent dan wat aangegeven staat in de documentatie van Moodle[17], omwille van het klein aantal actieve gebruikers op dit moment. In de toekomst kan deze instelling eenvoudig worden aangepast. De gebruikte configuratie en werkwijze is gedocumenteerd in de technische appendix A (zie A.2).

3.4.1.5 Back-up

De twee datavolumes (MariaDB en Moodle) worden dagelijks geback-upt aan de hand van een `cron`-taak die een back-up-script oproept. Dit back-upscript is ook terug te vinden in de technische appendix A (zie A.1.5). De back-ups worden gedurende zeven dagen bewaard op dezelfde machine, zodat Co-De in geval van nood hersteld kan worden.

3.4.2 Beveiliging

In Moodle
Moodle tracht de applicatie zo veel mogelijk te beveiligen door een juiste configuratie. Daarvoor zijn er richtlijnen opgesteld[18] die ook bij Co-De gevolgd worden.

HTTPS
Co-De ondersteunt `HTTPS`, zodat een veilige uitwisseling van gegevens mogelijk is. Het `HTTPS`-verkeer op Co-De verloopt volgens poort `443` (standaard). Een gebruiker die de site via `HTTP` tracht te bereiken (poort `80`), wordt automatisch doorgestuurd naar de `HTTPS`-versie.

Het gesigneerde `SSL`-certificaat, dat nodig is om `HTTPS` mogelijk te maken, wordt periodisch aangeleverd door "Let's Encrypt"[19]. Om gebruik te kunnen maken van Let's Encrypt moet men kiezen voor een compatibele *client*. Omdat er geen rechten voorzien zijn om software te installeren voor `code@code.experiments.cs.kuleuven.be`, werd opnieuw geopteerd voor een Let's Encrypt *client* die beschikbaar is als een Docker-*image*, namelijk "Zerossl"[20]. De geldigheidsduur van een `SSL`-certificaat is beperkt; voor het Zerossl-certificaat is dat twee maanden. Daarom is het van belang om tijdig een nieuw certificaat te genereren met de Zerossl-container. De richtlijnen hiervoor zijn terug te vinden in de technische appendix A (zie A.9).

[16] https://docs.moodle.org/24/en/Cron.
[17] Moodle geeft aan om het script elke minuut te laten uitvoeren.
[18] https://docs.moodle.org/34/en/Security_recommendations.
[19] https://letsencrypt.org/.
[20] https://hub.docker.com/r/zerossl/client/.

3.4.3 Technische vereisten verbonden aan lesinhoud

3.4.3.1 Jobe server voor programmeeropdrachten via CodeRunner

Het is mogelijk om programmeeropdrachten te integreren in lessen op Co-De. Dit is geen basisfunctionaliteit van Moodle en daarom moet er een extra plugin geïnstalleerd worden. Op Co-De wordt gebruikgemaakt van de plugin "CodeRunner"[21]. CodeRunner gedraagt zich als een bijkomend vraagtype dat kan worden geselecteerd tijdens het ontwerp van een test (zie 3.5.2). De programmeeropdrachten kunnen in een hele reeks programmeertalen opgelost worden. De oplossingen van gebruikers worden dan uitgevoerd op een hiervoor bestemde server. CodeRunner ondersteunt voorlopig alleen de "Jobe" server. Er is een standaard Jobe server die mag gebruikt worden voor het testen van de installatie, maar voor een publiek platform met grotere capaciteit is een eigen Jobe server noodzakelijk[22]. Jobe is eveneens beschikbaar als Docker-*image*[23] en dus werd Jobe als extra container toegevoegd aan de architectuur van Co-De (zie rechts op figuur 3.1 op p. 31). Meer gedetailleerde informatie over het opzetten van Jobe is terug te vinden in appendix A (zie A.4).

3.4.3.2 Hulpbronnen en mediabestanden

De hulpbronnen en mediabestanden uit de lessen op Co-De zijn op dezelfde server geplaatst als de Moodle-applicatie zelf met behulp van `SFTP`. Zo zijn ze beschikbaar vanop hetzelfde domein (`$code`) via `HTTPS` en worden ze door de verschillende browsers onmiddellijk als betrouwbare inhoud aanvaard. Dit is vooral van belang om vlot te interageren met de animaties (zie ook 3.5.2.2). Figuur 3.2 toont een hoog-niveau overzicht van de beschikbare hulpbronnen en multimedia op Co-De.

- ▷ `resources`
  - ▷ `paardenronde`: afbeeldingen en animaties voor de les.
  - ▷ `doolhoven`: afbeeldingen en animaties voor de les.
  - ▷ `marslander`: afbeeldingen en animaties voor de les.
  - ▷ `emoji`: emoticons voor de volledige site.
  - ▷ `processing`: hulpbronnen voor de animaties op Co-De.
  - ▷ `programmeeropdrachten`: hulpbestanden voor programmeeropdrachten.
    - ▷ `doolhoven`
  - ▷ `lessen`: lesbestanden (`.mbz`) om in te laden in een eigen Moodle-site.
  - ▷ `progress_bar.zip`: plugin met voortgangsbalk.
  - ▷ `relative_completion.zip`: plugin met relatieve voltooiing.
  - ▷ `theme_code.zip`: plugin met het thema "code".

Figuur 3.2: Een hoog-niveau overzicht dat de hiërarchie voor hulpbronnen en mediabestanden op Co-De toont.

[21] http://coderunner.org.nz/.
[22] http://coderunner.org.nz/mod/book/view.php?id=198&chapterid=800.
[23] https://hub.docker.com/r/trampgeek/jobeinabox/.

3.5 Ontwerp en implementatie

In dit deel gaan we dieper in op het ontwerp en onze implementatie van Co-De. Onder het ontwerp vallen de vereisten waaraan het leerplatform moet voldoen. Dit kunnen zowel technische als inhoudelijke vereisten zijn. Sommige van deze vereisten zijn al standaard ingebouwd in Moodle, andere nog niet. Onze implementatie omvat het toevoegen van deze extra vereisten die niet in Moodle 3.4.1 beschikbaar zijn.

Wat volgt is een functionele opdeling van de vereisten voor het leerplatform. Bij elk onderdeel wordt aangehaald wat de verwachte functionaliteit is en of deze is ingebouwd in Moodle. Bij functionaliteiten waarvoor dit niet het geval is, wordt dieper ingegaan op de aanpassingen of toevoegingen die werden gedaan voor Co-De.

3.5.1 Lessen

De lessen op Co-De zijn een van de belangrijkste bouwstenen van het leerplatform. Een les bestaat uit een reeks activiteiten die zijn ondergebracht in secties. Door de opeenvolgende activiteiten af te werken (pagina's met uitleg, opdrachtjes, animaties, meerkeuzevragen, programmeeropdrachten, reflectiemomenten...), doorloopt een leerling de les. Elke les bevat één of meerdere probleemstellingen en doorheen de les bedenkt men hiervoor een oplossing. Daarbij wordt steeds getraind op enkele kernelementen van computationeel denken (zie 2.3).

Binnen de context van Co-De is het belangrijk dat de lessen kunnen worden aangepast aan de situatie waarin ze worden gebruikt. Elke klasgroep, school en leerkracht is anders en de voorkennis, het doel en de beschikbare tijd kan erg verschillen. In dit deel van de tekst worden de vereisten en implementatie toegelicht om dat mogelijk te maken. In onderdeel 4.2.4 wordt toegelicht hoe de lessen kunnen worden aangepast in Co-De.

Moodle biedt al een systeem aan met cursussen[24] dat, buiten de benaming, nauw aansluit bij het concept van lessen in Co-De. Dit systeem met cursussen wordt integraal gebruikt in Co-De en de benaming cursus werd behouden aangezien ze de alom gekende norm is in de Nederlandstalige Moodle. In deze tekst zullen we echter spreken over lessen.

3.5.1.1 Lesstructuur

De lessen zijn steeds opgebouwd volgens eenzelfde structuur. Deze vaste structuur biedt een duidelijk kader voor zowel de leerkracht als de leerling om de lessen op Co-De te gebruiken. Ondanks die vaste structuur is het mogelijk om de lessen eenvoudig aan te passen voor een specifieke situatie of klasgroep (zie 4.2.4).

[24] https://docs.moodle.org/34/en/Courses.

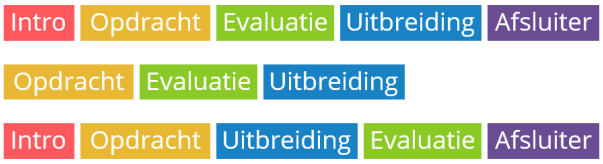
Informatie voor leerkrachten
Elke les start met informatie die gericht is aan de leerkracht. Die info bestaat uit vier vaste delen (zie figuur 3.3). De leerlingen kunnen dit gedeelte niet zien.

Figuur 3.3: De sectie met informatie voor leerkrachten, terug te vinden bovenaan in iedere les.

1. *Uitleg over de les*: In dit onderdeel wordt de les beschreven voor de leerkracht. De verschillende lesonderdelen worden toegelicht en het doel van de les wordt duidelijk gemaakt. Er wordt ook dieper ingegaan op welke elementen van computationeel denken in de les aan bod komen, waarom en hoe. Ook andere concepten en begrippen die belangrijk of nieuw zijn, worden toegelicht.

2. *Verschillende leerpaden*: Hier worden een aantal gangbare leerpaden aangereikt waaruit de leerkracht inspiratie kan putten om de les aan te passen op maat van zijn of haar klasgroep. Een leerpad doelt op een bepaalde volgorde van activiteiten binnen de les, waarbij ook bewust activiteiten kunnen worden weggelaten. Figuur 3.4 geeft een voorbeeld van drie verschillende leerpaden met maximaal vijf activiteiten weer.

   Bij het aanreiken van de verschillende leerpaden wordt ook een indicatie gegeven van welk leerpad mogelijk beter aansluit bij bepaalde leerlingen of groepen. Er wordt dieper ingegaan op het aanpassen van een les in onderdeel 4.2.4.



Figuur 3.4: Een voorbeeld van drie verschillende leerpaden. De balkjes stellen de activiteiten voor en een rij van balkjes vormt een leerpad.

3. *Referenties & hulpbronnen*: Onder referenties en hulpbronnen worden alle gebruikte bronnen vermeld. Zowel de oorsprong van de afbeeldingen, waar het les-idee vandaan komt, als de gebruikte tools en programma's voor het maken van de illustraties en animaties worden hier besproken.

4. *Les aanvragen*: Dit onderdeel bestaat voornamelijk uit een formulier om de les aan te vragen om er mee aan de slag te gaan in een klascontext. Je kan hier aangeven welke les(sen) je als leerkracht wil gaan gebruiken, met welk doel en met hoeveel leerlingen. Een beheerder van Co-De zal dan een kopie van de *demo-lessen* voor die leerkracht opzetten. In 3.5.1.3 wordt dieper ingegaan op het verschil tussen *demo-lessen* en *eigen lessen*.

   Naast het formulier om de les aan te vragen, vind je hier ook een `.mbz`-bestand van de les. Dit bestand kan gebruikt worden om de les in te laden op een eigen Moodle-site (zie 3.6.2.1).

Lesspecifieke inhoud

Na de informatie voor leerkrachten volgt een deel lesspecifieke inhoud. Klassiek bevat deze een probleemstelling (als inleiding) gevolgd of begeleid door teksten, afbeeldingen en animaties. In dit onderdeel van de les worden vaak vragen en keuzes aan de leerlingen voorgelegd en is er ruimte voor overleg in groepjes of met de hele klas. In sommige lessen komen ook programmeeropdrachten en testjes aan bod. De lesspecifieke inhoud vormt het grootste deel van de les en varieert het meest van les tot les. In figuur 3.5 wordt bijvoorbeeld de lesspecifieke inhoud getoond uit de les de drie bekers.

Figuur 3.5: Het lesoverzicht van de drie bekers. De verschillende activiteiten zijn ondergebracht in onderwerpen (secties).

Computationeel denken

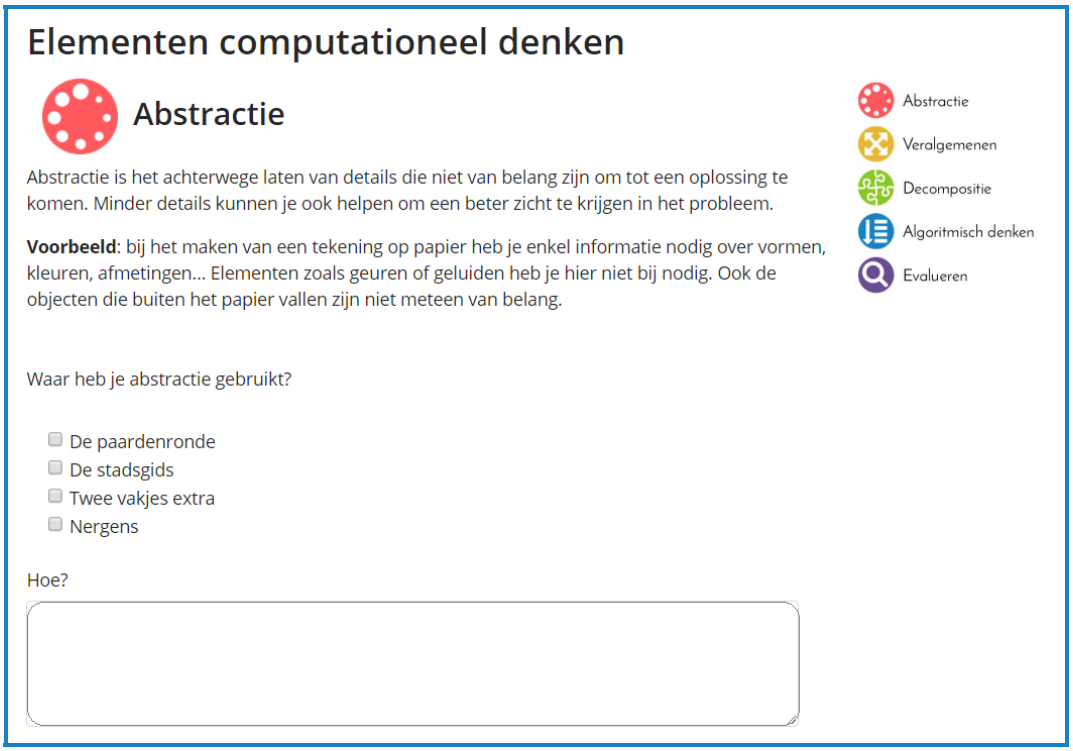
Dan volgt een gedeelte dat expliciet ingaat op het computationeel denken in de vorm van een vragenlijst. De vijf elementen van CD worden één voor één toegelicht aan de hand van eenvoudige voorbeelden. De leerling wordt hierbij gevraagd om te beschrijven waar hij of zij binnen deze les de verschillende elementen aan bod heeft zien komen (zie figuur 3.6). Na het antwoorden op deze vraag wordt per element ook vermeld waar het element vooral naar voren kwam. Deze vragenlijst is een licht aangepaste feedback-activiteit uit Moodle[25] waarvan een herbruikbaar sjabloon[26] wordt voorzien voor nieuwe lessen.

[25] https://docs.moodle.org/34/en/Feedback_activity.
[26] https://docs.moodle.org/34/en/Feedback_templates.

Uitbreidingen en variaties
Binnen de meeste lessen is daarna nog ruimte voor uitbreidingen en variaties. Dit zijn vaak open probleemstellingen zonder dat het antwoord stap voor stap wordt toegelicht. Wel worden bij elke uitbreiding of variatie enkele handvatten aangereikt die belangrijk zijn in de oplossing. Dit deel van de les zorgt dat ook de snelle werkers hun aandacht niet verliezen en voldoende worden uitgedaagd. Veel van deze uitbreidingen of variaties bieden inspiratie voor het uitbouwen van een nieuwe les.

Afsluiter
De les wordt doorgaans afgesloten met een weetje en een begeleidende afbeelding.

Deze lesstructuur kan worden bekomen via het sectie- en activiteitensysteem[27] dat aanwezig is in Moodle. De verschillende grote delen worden ondergebracht in secties in het onderwerpformaat[28] waardoor ze een toepasselijke naam kunnen krijgen. Het gedeelte met lesspecifieke inhoud kan daarbij meerdere secties bestrijken. De eerste sectie met informatie voor de leerkrachten is via reeds ingebouwde functionaliteiten[29] verborgen voor de leerlingen.



Figuur 3.6: De eerste vraag uit het feedbackformulier rond CD aan het einde van de les paardenronde & stadsgids.

[27] https://docs.moodle.org/34/en/Courses.
[28] https://docs.moodle.org/34/en/Course_formats#Topics_format.
[29] https://docs.moodle.org/20/en/Course_sections#Hiding_sections.

3.5.1.2 Lesverloop

Het doel van dit leerplatform is om een leervorm aan te bieden waarbij op een interactieve manier wordt geleerd en iedere leerling op zijn of haar eigen tempo kan werken. De les vindt plaats binnen een klassikale context waarin ook de leerkracht een actieve rol speelt. De opstelling zelf kan op verschillende manieren: in één of meerdere (klas)lokalen, individueel of in groepjes aan de computer of zelfs klassikaal met behulp van een smartboard.

De lessen op het platform moeten zodanig opgebouwd zijn dat de leerlingen er zelf mee aan de slag kunnen en zelfstandig kunnen werken. De lesinhoud is hierin de belangrijkste factor (zie hoofdstuk 4), maar ook de opbouw van het platform speelt hierin een rol. Het platform moet namelijk zo zijn opgebouwd dat een logische navigatie en een duidelijk lesverloop mogelijk zijn.

Navigatie
Zoals al werd beschreven bestaan de lessen uit verschillende onderdelen (activiteiten) die ondergebracht worden in onderwerpen (secties). De leerlingen volgen de les door opeenvolgende activiteiten af te werken. Een overzicht van de verschillende soorten activiteiten volgt in 3.5.2. In Moodle hebben activiteiten en secties al een standaard volgorde, namelijk de volgorde van boven naar onder in het lesoverzicht (afgebeeld op figuur 3.5). Ook kan men steeds van de huidige activiteit doorklikken naar de vorige en volgende activiteit of via een selectiemenu springen naar andere activiteiten. De basis navigatie-functionaliteit is dus al aanwezig. Uit onze testen bleek echter dat de testgebruikers de navigatie niet intuïtief vonden (zie 3.7). Daarom zijn enkele zaken aangepast of toegevoegd aan de standaard Moodle-functionaliteiten.

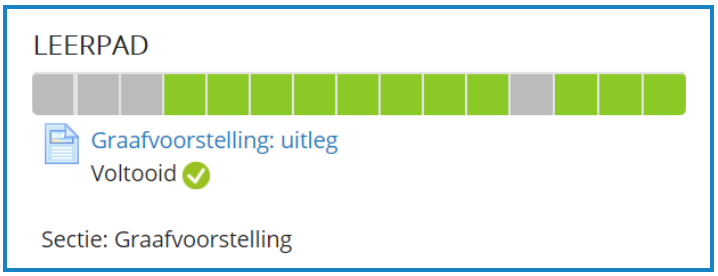
De navigatie tussen de activiteiten is vereenvoudigd door een ingreep in het thema. De steeds wisselende en uitgebreide navigatiemogelijkheden onderaan elke activiteit zijn omgezet in een coherente en eenvoudige navigatiebalk. Onderaan elke activiteit kan men kiezen tussen "◁ Vorige" of "Volgende ▷" zodat het sequentiële verloop van de les wordt benadrukt. Een tweede wijziging is dat de link naar de volgende activiteit pas zichtbaar wordt wanneer de huidige activiteit voltooid is. Deze aanpassingen kaderen in een reeks stijl-aanpassingen voor Co-De en ze zitten vervat in een standaard Moodle-thema dat ook losstaand van Co-De beschikbaar is[30]. Deze aanpassingen zijn voornamelijk terug te vinden in de `.scss`-bestanden van het thema. Meer technische informatie hierover is terug te vinden in appendix A (zie A.8).

[30] `$code/resources/theme_code.zip`.

Voortgangsbalk

Een tweede tekortkoming aan de navigatie is dat de leerlingen geen overzicht hebben van welke activiteiten ze al hebben afgerond. Dit is vooral belangrijk wanneer leerlingen opdrachten moeten insturen of vragen moeten beantwoorden. Vertrekkend van de bestaande Moodle-plugin "Completion Progress"[31] is daarom een voortgangsbalk toegevoegd bovenaan het lesoverzicht (zie figuur 3.7). Hier kan de leerling te allen tijde terecht om zijn of haar voortgang binnen de les na te kijken. Enkel de activiteiten die toegankelijk zijn voor de leerling worden getoond in de balk en afgewerkte activiteiten worden in het groen gemarkeerd.

De veranderingen om de voortgangsbalk in zijn huidige vorm te tonen, bevinden zich op verschillende niveaus. Ten eerste wordt bij elke activiteit in de voortgangsbalk de bijbehorende sectie getoond. Hiervoor werd een extra methode geschreven in de bestaande `lib.php`[32] van de plugin. Ten tweede werd de plugin naar het Nederlands vertaald. In Moodle worden taalpakketten steeds eenvoudigweg in afzonderlijke bestanden toegevoegd in de bijhorende `lang`-map[33] van een plugin. Ten derde is ook de positie van de voortgangsbalk aangepast. De oorspronkelijke plugin is een `block`-plugin, wat betekent dat hij standaard in een zijbalk van een Moodle-pagina wordt weergegeven. Om de voortgangsbalk centraal weer te geven, is een aanpassing van het thema vereist. In `config.php`[34] is er een extra optie voorzien om blokken centraal te plaatsen. Daarnaast moet telkens gecontroleerd worden of er blokken centraal moeten worden weergegeven in de lay-out-bestanden (voor Co-De uitsluitend `columns2.php`[35]) en bijhorende Mustache templates (namelijk `columns2.mustache`[36]). De zonet beschreven wijzigingen in de broncode zijn terug te vinden in appendix A (zie A.7).

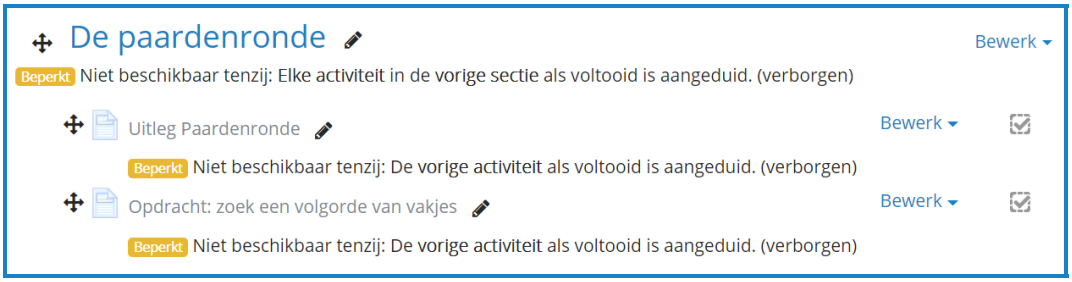


Figuur 3.7: Een momentopname van de voortgangsbalk bij de drie bekers.

[31] https://moodle.org/plugins/block_completion_progress.
[32] **bestand**: moodle\blocks\completion_progress\lib.php.
[33] **map**: moodle\blocks\completion_progress\lang\.
[34] **bestand**: moodle\theme\code\config.php.
[35] **bestand**: moodle\theme\code\layout\columns2.php.
[36] **bestand**: moodle\theme\broncode\templates\columns2.mustache.

Relatieve voltooiing
Het is bij het doorlopen van de les niet de bedoeling dat de leerlingen al de hele les kunnen zien. Titels van toekomstige activiteiten kunnen bijvoorbeeld te veel prijsgeven of men kan de oplossing op bepaalde vragen al bekijken nog voor de vragen gesteld zijn. Het is doorgaans de bedoeling dat activiteiten en secties pas zichtbaar worden wanneer de vorige activiteit of sectie volledig is afgerond.

Via de "beperk toegang" opties van Moodle kan dit gedrag bekomen worden door bij elke activiteit als voorwaarde te stellen dat die niet zichtbaar mag worden tenzij de vorige activiteit is voltooid[37]. Om dit te doen moet de vorige activiteit gekend zijn en worden geselecteerd uit een lijst van alle activiteiten. Dit soort absolute relatie is echter niet handig voor een online leerplatform waarbij de inhoud zeer aanpasbaar moet zijn; ook de volgorde van de lesonderdelen. Met dit systeem komt men immers in de problemen wanneer men zegt dat activiteit B pas zichtbaar mag worden wanneer activiteit A volledig is afgerond en men later activiteit A en B van plaats wisselt of activiteit A weglaat. Om deze bewerkingen wel zonder verdere aanpassingen te kunnen uitvoeren zullen we een systeem uitbouwen waarin activiteiten niet absoluut, maar relatief aan elkaar gelinkt worden.



Figuur 3.8: Een voorbeeld van relatieve voltooiing in de paardenronde & stadsgids. Alleen de leerkrachten en beheerders kunnen de voorwaarden aflezen in het lesoverzicht. De volgorde van de secties of activiteiten kan eenvoudig aangepast worden door de onderdelen te verslepen via de afgebeelde pijlen (links). De eventuele voorwaarde dat de vorige activiteit of sectie voltooid moet zijn, blijft nog steeds van kracht. De toevoeging (verborgen) aan de voorwaarde duidt erop dat het onderdeel pas zichtbaar zal zijn als aan de voorwaarde voldaan is. Dit in tegenstelling tot wanneer het onderdeel al zichtbaar zal zijn in het lesoverzicht, maar nog niet toegankelijk is.

In dit nieuwe systeem kan men zeggen dat een activiteit pas zichtbaar en/of beschikbaar wordt wanneer de *vorige* activiteit is voltooid. De *vorige* activiteit is dan afhankelijk van waar de activiteit op dat moment staat. Op deze manier kunnen activiteiten A en B zonder problemen van plaats worden gewisseld of worden weggelaten (figuur 3.8). Onze implementatie houdt ook rekening met gevallen waar er geen vorige (zichtbare) activiteit bestaat.

[37] https://docs.moodle.org/34/en/Restrict_access_settings#Activity_completion.

Hetzelfde systeem is ook beschikbaar voor secties, hierbij kan gekozen worden om pas zichtbaar te worden wanneer *minstens één* of *alle* activiteiten uit de *vorige* sectie afgewerkt zijn.

Deze functionaliteit is geïmplementeerd in de vorm van een standaard Moodle-plugin. Alle lessen op Co-De gebruiken dit systeem van toegangsbeperkingen voor de verschillende lesonderdelen. Deze plugin kan men ook installeren op een eigen Moodle-site. Dit wordt meer in detail toegelicht in 3.6.2.2 en in appendix A. Figuur 3.9 toont de samenstelling van de plugin relatieve voltooiing.

- ▷ `relative_completion`
  - ▷ `classes`: de plugin-logica.
    - ▷ `condition.php`: de logica die bepaalt of een specifieke activiteit of sectie zichtbaar en/of toegankelijk zal zijn.
    - ▷ `frontend.php`: de logica die bepaalt welke opties de gebruiker (leerkracht, cursusbeheerder) kan selecteren voor een bepaalde activiteit of sectie.
  - ▷ `lang`: de verschillende taalpakketten die ondersteund worden door de plugin.
  - ▷ `tests`: de testen voor de plugin-logica.
  - ▷ `yui`: het formulier dat de beheerder van de les te zien krijgt bij het aanpassen van een activiteit of sectie. Dit beeldt de opties uit `frontend.php` af.
  - ▷ `version.php`: noodzakelijke meta-info over de plugin.

Figuur 3.9: Een hoog-niveau overzicht van de structuur van de plugin relatieve voltooiing. De hiërarchie is conform aan de opgelegde structuur van Moodle-plugins.

De voortgangsbalk en relatieve-voltooiing-voorwaarden zorgen voor een gestructureerde les waarin toch nog ruimte is voor flexibiliteit. De leerling kan op eigen tempo de les volgen en zijn voortgang bijhouden. De leerkracht kan op zijn beurt de volgorde van lesonderdelen aanpassen, zonder dat andere voorwaarden of configuraties moeten worden aangepast.

3.5.1.3 Soorten

In Co-De bestaan er, op enkele uitzonderingen na, twee soorten lessen: *demo-lessen* en *eigen lessen*. De *demo-lessen* zijn standaard lessen die iedereen kan volgen, ze hebben namen als “De Marslander” of “Doolhoven”. *Eigen lessen* zijn lessen die zijn verbonden aan een specifieke leerkracht of school. Deze kunnen worden aangepast door de leerkracht en de antwoorden van de leerlingen zijn zichtbaar voor de leerkracht. *Eigen lessen* krijgen in hun naam een voorvoegsel dat verwijst naar de school en de klasgroep zoals bijvoorbeeld “Dia7SBT - Paardenronde & Stadsgids”.

Demo-lessen
De *demo-lessen* zijn de standaard lessen zoals ze zijn opgebouwd door de ontwikkelaar van de les in een basis-configuratie en met een vast leerpad. Iedereen kan zichzelf op deze lessen inschrijven en deze doorlopen als leerling of demo-leerkracht (zie 3.5.3). Deze registratie is steeds 30 dagen geldig, en kan hernieuwd worden zonder de gemaakte voortgang te verliezen. Aangezien deze lessen de standaardlessen zijn, kunnen ze niet worden aangepast voor eigen gebruik. De volgorde van de lesonderdelen, het leerpad en de zichtbare delen zijn al vastgelegd door een beheerder. Er zijn ook geen persoonsgegevens zichtbaar, wat betekent dat een demo-leerkracht de antwoorden van leerlingen niet kan zien of beoordelen.

Demo-lessen zijn bedoeld om snel mee aan de slag te kunnen gaan of om als leerkracht te kunnen zoeken naar geschikte lessen. Ook leerlingen kunnen, na registratie op Co-De, meteen aan de slag met deze lessen zonder enige tussenkomst van een leerkracht.

Alle *demo-lessen* zijn terug te vinden op Co-De onder de cursuscategorie "Demolessen Computationeel Denken" via `$code/course/index.php?categoryid=2`.

Eigen lessen
Daarnaast zijn er *eigen lessen* op Co-De. Wanneer een leerkracht wijzigingen wil aanbrengen in een les en/of de antwoorden van de leerlingen wil zien, kan hij de les aanvragen via het formulier in het deel "informatie voor de leerkrachten" van een *demo-les* (zie 3.5.1.1). Een beheerder van Co-De zal dan een nieuwe les toevoegen met de lesinhoud uit de *demo-les* en de leerkracht instellen als leerkracht van die les. Vanaf nu kan de leerkracht deze les zelf beheren, aanpassen en uitbreiden. De leerkracht kan ook beheren welke gebruikers toegang hebben tot de les en met welke rol. Het is niet mogelijk voor gebruikers om zichzelf aan te melden op *eigen lessen*, tenzij de leerkracht dit zo ingesteld heeft.

De leerkracht kan nu een leerpad kiezen, lesonderdelen herordenen, verbergen of toevoegen en leerlingen inschrijven op de les. Onderdeel 4.2.4 licht de verschillende mogelijkheden om lessen aan te passen toe. Uiteraard kan een leerkracht ook beslissen om de les niet aan te passen, maar wel kiezen voor een *eigen les* om louter de antwoorden van zijn of haar klasgroep(en) te zien.

Alle *eigen lessen* worden onderverdeeld per school en zijn terug te vinden op Co-De onder de cursuscategorie "Scholen" via `$code/course/index.php?categoryid=3`.

Samenvattend bepalen de cursuscategorieën[38] op Co-De het onderscheid tussen de twee soorten lessen (*demo-lessen* en *eigen lessen*). Elke gebruiker beschikt over die categorieën om snel een les terug te vinden. Op het vlak van de rollen is er ten slotte een onderscheid tussen de gewone rollen (voor *eigen lessen*) en de nieuwe rol "demo-leerkracht" (voor *demo-lessen*). Een overzicht van de verschillende rollen en hun rechten is terug te vinden in tabel 3.2 op p. 57.

[38] `https://docs.moodle.org/34/en/Course_categories`.

3.5.2 Activiteiten

In de lessen op Co-De komen verschillende soorten activiteiten aan bod. Elke soort vertrekt van een activiteit[39] of bron[40] die beschikbaar is in Moodle 3.4 of van een toegevoegde plugin[41]. Moodle biedt nog heel wat meer activiteitensoorten aan, maar deze worden niet gebruikt in de uitgebouwde lessen.

Binnen Co-De zijn enkele wijzigingen aangebracht om het gedrag van de activiteiten bij te sturen voor het beoogde gebruik. Hierna wordt dieper ingegaan op de meest gebruikte soorten activiteiten en waarom deze zinvol zijn. Ook de aanpassingen die zijn doorgevoerd voor Co-De ten aanzien van het standaardgedrag worden belicht.

Hoe men deze activiteiten kan gebruiken bij het aanpassen van de lessen of bij het bouwen van nieuwe lessen komt aan bod in 3.6 en wordt verder toegelicht in de gebruikershandleiding[42]. Enkel door activiteiten correct te configureren kan men het gedrag zoals het hieronder wordt beschreven afdwingen in Co-De.

3.5.2.1 Pagina

Een pagina[43] is een eenvoudige en multifunctionele activiteit. Pagina's bestaan enkel uit een titel en een inhoud die `html` ondersteunt. Ze worden gebruikt voor de weergave van tekst, afbeeldingen, geluids- of videofragmenten, wiskundige formules en voor de insluiting van `html`-inhoud.

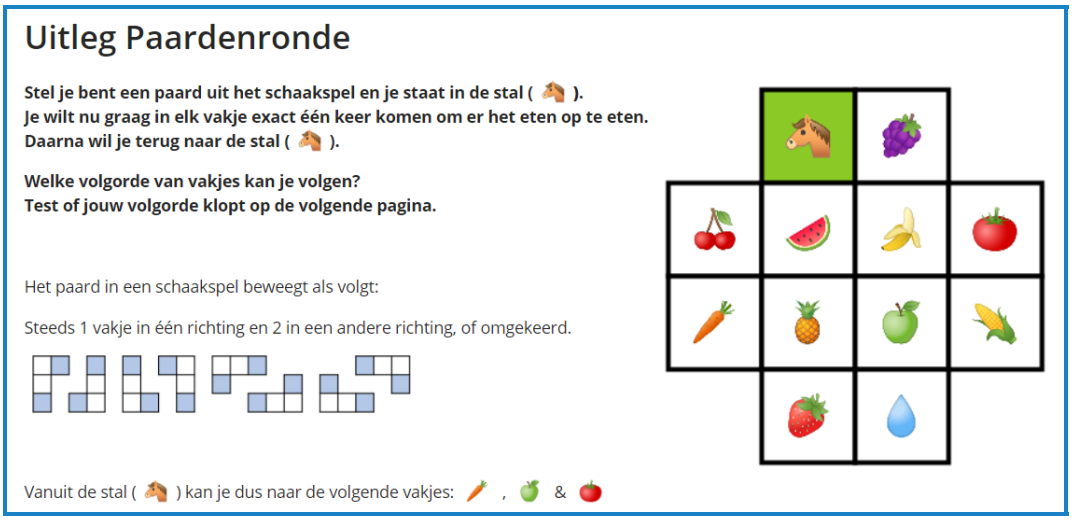
Figuur 3.10: De bewerkingsbalk voor het bewerken van inhoud uit velden die html ondersteunen.

[39] https://docs.moodle.org/34/en/Activities.
[40] https://docs.moodle.org/34/en/Resources.
[41] https://moodle.org/plugins/browse.php?list=category&id=1.
[42] $code/handleiding.
[43] https://docs.moodle.org/34/en/Page_resource.

Aan de pagina-functionaliteit uit Moodle is niets gewijzigd in Co-De. Wel zijn enkele toevoegingen en aanpassingen doorgevoerd aan de bewerkingsbalk die bovenaan het inhoudelijk gedeelte staat (zie figuur 3.10). Ten eerste is het Courier-font[44] toegevoegd om programmacode een vertrouwde weergave te geven. Ten tweede zijn ook de kleuropties aangepast aan het kleurenschema van Co-De (zie figuur 3.11). Zo kan men tekst kleuren of markeren in de vijf terugkerende kleuren. Deze toevoegingen zijn ook zinvol voor andere activiteitensoorten aangezien dezelfde bewerkingsbalk overal gebruikt wordt.

Voorbeelden van pagina's zijn de delen met uitleg voor de leerkracht (zie 3.5.1.1), pagina's met uitleg en afbeeldingen of de probleemstelling aan het begin van de les (zie afbeelding 3.12).

Figuur 3.11: De vijf kleuren van Co-De met hun hexadecimale waarde.



Figuur 3.12: Een voorbeeld van een pagina-activiteit met tekst en afbeeldingen.

[44] https://nl.wikipedia.org/wiki/Courier_(lettertype).

3.5.2.2 Animatie

Animaties zijn pagina's waarop een interactieve animatie is ingesloten. Het kan gaan om een opdracht waarbij een bepaald probleem moet worden opgelost. De leerlingen kunnen hun oplossingen dan nakijken en krijgen specifieke feedback binnen de animatie. Hierbij kunnen verschillende versies worden uitgebouwd met elk een verschillend niveau van ondersteuning. De leerkracht kan dan kiezen welke versie het meest geschikt is voor zijn of haar klasgroep. Enkele voorbeelden hiervan zijn:

Binnen de les paardenronde & stadsgids:
- De opgave bij de paardenronde[45]
- Dezelfde opgave met meer ondersteuning[46]

Binnen de les de marslander:
- De opgave bij de marslander met een veld van 4 op 4 meter[47]
- De opgave bij de marslander met een veld van 6 op 6 meter[48]

Een tweede gebruik is de visualisatie en dynamische weergave van een bepaald concept dat zonder interactieve mogelijkheden moeilijk valt te beschrijven. Een voorbeeld:

Binnen de les paardenronde & stadsgids:
- Stapsgewijze omzetting van een complex probleem in een graaf[49]

De animaties zijn gemaakt in "Processing"[50], een softwarepakket om visueel te programmeren in Java syntax. Elke animatie is een apart project bestaande uit één of meerdere `.pde`-bestanden die in een map worden geplaatst bij de hulpbronnen (zie 3.4.3.2). Om lokaal ontwikkelde animaties online weer te geven wordt "Processing.js"[51] gebruikt. Via een `.html`-bestand kan de animatie nu online worden gebruikt. Figuur 3.13 geeft de structuur van dit `.html`-bestand weer. Eens de animatie zichtbaar is op een webpagina kan ze via een `iframe` worden ingesloten in een pagina-activiteit (zie figuur 3.14). Hierbij wordt ook steeds een link naar de oorspronkelijke animatie vermeld voor leerlingen bij wie de animatie niet zichtbaar is (zie "Indien de opdracht niet laadt..." in figuur 3.14).

[45] `$code/resources/paardenronde/opgave1/`.
[46] `$code/resources/paardenronde/opgave1_hulp/`.
[47] `$code/resources/marslander/opgave_4m/`.
[48] `$code/resources/marslander/opgave_6m/`.
[49] `$code/resources/paardenronde/magie/`.
[50] `https://processing.org/`.
[51] `http://processingjs.org/`.

```
<!DOCTYPE html>
<html>
<head>
    <!-- locate processing.js source files -->
    <script
     src="https://code.experiments.cs.kuleuven.be/resources/processing/processing.js">
    </script>
    <script
     src="https://code.experiments.cs.kuleuven.be/resources/processing/processing.min.js">
    </script>
</head>
<body style="margin:0px;">
    <!-- locate .pde files containing animation, separated by blank spaces -->
    <canvas width="1960" height="500"
        data-processing-sources="file1.pde file2.pde file3.pde">
    </canvas>
</body>
</html>
```

Figuur 3.13: De html-template voor het tonen van een animatie die ontwikkeld is in Processing.

```
<iframe
    src="https://code.experiments.cs.kuleuven.be/resources/LES/ANIMATIE"
    width="955px" style="height:534px;
    margin:0px; border:2px solid black;">
</iframe>
<br>
Indien de opdracht niet laadt, klik dan
<a href="https://code.experiments.cs.kuleuven.be/resources/LES/ANIMATIE" target="_blank">
    hier
</a>.
```

Figuur 3.14: Het iframe voor het embedden van een animatie die ontwikkeld is in Processing.

3.5.2.3 Keuze

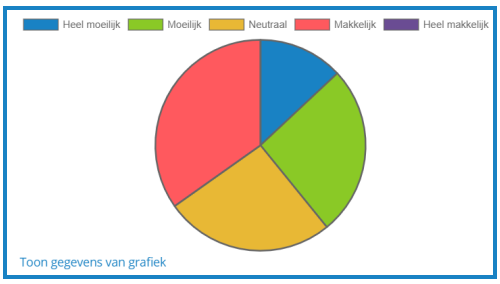
In een keuze-activiteit kunnen of moeten de leerlingen antwoorden op een meerkeuzevraag. Het vraaggedeelte is een veld dat `html` ondersteunt, net zoals bij pagina's (zie 3.5.2.1). De mogelijke antwoorden bestaan uit korte stukjes ongeformatteerde tekst waarbij kan gekozen worden om één of meerdere antwoorden toe te laten per leerling. Geen van deze antwoorden is juist of fout; of: het gaat hier eerder om een systeem om te stemmen dan om te testen. De leerkracht kan ervoor kiezen om, nadat de leerlingen een keuze maakten, alle antwoorden aan de leerlingen te tonen in een taartdiagram. Dit kan nuttig zijn voor de leerling om zijn of haar keuze te plaatsen binnen de klasgroep of om een bespreking op gang te brengen. Een voorbeeld van een keuze-activiteit en de weergave van de antwoorden is te zien in figuur 3.15.

Deze activiteit is standaard ingebouwd in Moodle 3.4[52] en er is inhoudelijk niets aan gewijzigd voor Co-De. Enkel de weergave van de antwoorden is aangepast. De antwoorden worden in Moodle standaard weergegeven in een staafdiagram met héél

[52] https://docs.moodle.org/34/en/Choice_activity.

grote balken in dezelfde kleur. Deze weergave biedt echter weinig meerwaarde en is op vele schermen niet goed zichtbaar. Daarom is het grote staafdiagram vervangen door een kleiner taartdiagram dat verschillende kleuren gebruikt (zie figuur 3.15b). Dit gebeurt door het overschrijven van de functie `display_publish_anonymous` in de `renderer`[53] van de keuze-activiteit vanuit de `core_renderer`[54] in het eigen thema voor Co-De[55]. Daarnaast is de grootte van het diagram aangepast door het overschrijven van enkele `css`-attributen via de `.scss`-bestanden binnen het thema. Tot slot zijn de standaardkleuren vervangen door in het `config`-bestand van het thema de `$CFG->chart_colorset` te overschrijven. Op deze manier kunnen updates voor de keuze-activiteit en Moodle worden doorgevoerd, zonder deze aanpassing opnieuw te moeten doen (zie 3.4.1.3). Al deze aanpassingen worden in meer detail besproken in de technische appendix A.

(a) Een opgave waarbij de leerlingen één antwoord kunnen selecteren.



(b) Een overzicht van de antwoorden dat de deelnemers te zien krijgen na het maken van de keuze.

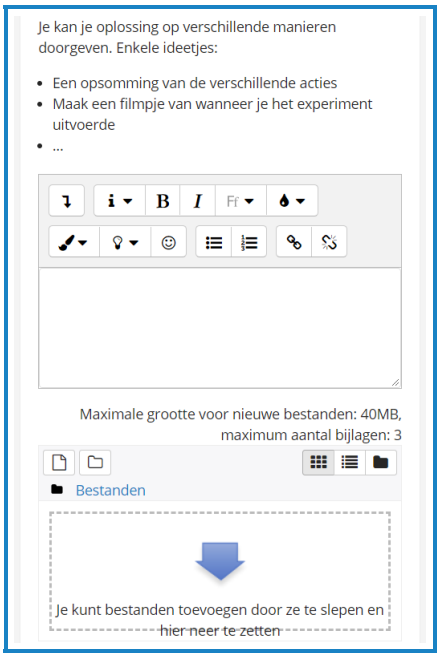
Figuur 3.15: Een voorbeeld van een keuze-activiteit.

[53] **bestand**: moodle\mod\choice\renderer.php.
[54] **bestand**: moodle\theme\code\classes\core_renderer.php.
[55] `$code/resources/theme_code.zip`.

3.5.2.4 Opdracht

Een opdracht in Moodle[56] bestaat uit twee delen: een beschrijving van de opdracht en ruimte om een antwoord in te dienen.

De beschrijving van de opdracht, klassiek een vraagstelling, is een veld dat `html` ondersteunt, net zoals een pagina (zie 3.5.2.1). Het antwoord kan ook bestaan uit een dergelijk `html` veld waarin de leerling zijn of haar antwoord kan ingeven en waar, desgewenst, ook bestanden kunnen worden opgeladen (zie figuur 3.16). De leerkracht kan hierbij een minimum en/of maximum aantal woorden opleggen. Deze activiteit biedt daarnaast ook mogelijkheden om de antwoorden van de leerlingen te beoordelen en feedback hieraan toe te voegen. Deze opties worden echter niet gebruikt binnen de *demo-lessen* op Co-De.



Figuur 3.16: Een voorbeeld van een opdracht met de mogelijkheid om een bestand toe te voegen.

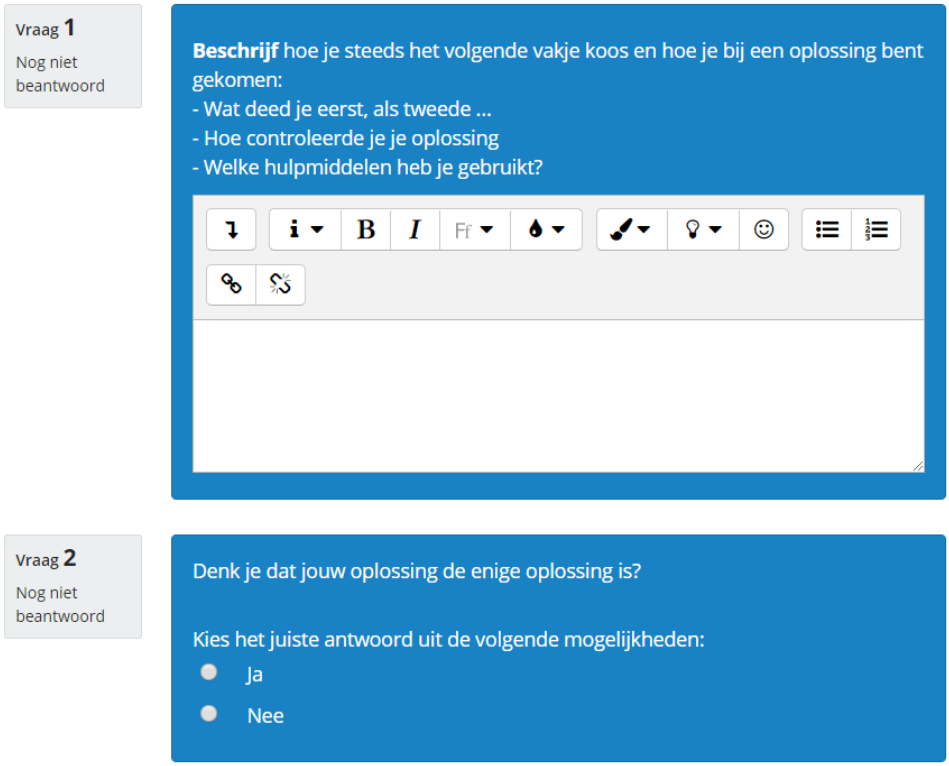
Functioneel is er aan deze activiteitensoort niets aangepast voor Co-De. Enkel visueel zijn een paar wijzigingen doorgevoerd door het verbergen van overbodige knoppen en tussenstappen. Deze kleine wijzigingen zijn terug te vinden in de `.scss`-bestanden binnen het thema[57].

Enkele voorbeelden van opdrachten zijn te vinden via deze snelkoppelingen:
- `$code/mod/assign/view.php?id=175`
- `$code/mod/assign/view.php?id=187`
- `$code/mod/assign/view.php?id=110`

[56] https://docs.moodle.org/34/en/Assignment_activity.
[57] $code/resources/theme_code.zip.

3.5.2.5 Test

Een test, ook wel quiz genoemd, is een activiteit die vele vormen kan aannemen. Testen bestaan steeds uit een reeks vragen verdeeld over één of meerdere pagina's (zie voorbeeld in figuur 3.17). Afhankelijk van het vraagtype kunnen bepaalde antwoorden juist, gedeeltelijk juist, of fout zijn en wordt dit gemeld aan de leerling. Aan elke vraag of specifiek antwoord kan ook feedback worden gekoppeld die wat meer uitleg geeft. Een test is dus als het ware een verzameling van vragen zoals een les een verzameling is van activiteiten en secties. Hij kan zeer kort (bijvoorbeeld slechts één vraag), maar ook zeer lang zijn (en grote delen van de les bevatten). Meer informatie over testen kan men terugvinden op de website van Moodle op `https://docs.moodle.org/34/en/Quiz_activity`. In de rest van deze sectie zullen we ons focussen op het gebruik en gedrag van testen binnen Co-De.



Figuur 3.17: Een voorbeeld van een test met twee vragen. De eerste vraag vereist een tekstueel antwoord dat manueel beoordeeld kan worden. De tweede vraag heeft een juist en een fout antwoord. De leerling zal na het indienen feedback krijgen over de correctheid van zijn of haar antwoord op die tweede vraag (zie bijvoorbeeld figuur 3.18 p. 52 voor feedback op een andere vraag).

Moodle bevat al een scala aan mogelijke vraagtypes[58] waaronder meerkeuzevragen, juist-of-fout-vragen, vragen met een vrije tekst als antwoord, drag-and-drop vragen, vragen met een numeriek antwoord... Daarnaast zijn er ook heel wat plugins die extra vraagtypes toevoegen[59] zoals het ordenen van elementen, positioneren van begrippen op afbeeldingen, plaatsen aanduiden op een kaart, wiskundige formules tekenen met de muisaanwijzer...

Binnen de test zijn er heel wat verschillende opties die het gedrag van de test beïnvloeden[60]. Geen van deze opties zorgt echter voor een vlot verloop van de testen zoals dat voor Co-De beoogd wordt. Het gewenste gedrag is immers dat bij het starten van de test, men meteen de eerste vragen krijgt voorgeschoteld en deze kan beantwoorden, men eenvoudig doorheen alle vragen kan navigeren en na het beantwoorden van de laatste vraag de feedback te zien krijgt. Dit sluit conceptueel nauw aan bij het sequentieel doorlopen van de les met de knoppen "◁ Vorige" en "Volgende ▷".

De grootste aanpassingen die gedaan werden op Co-De om het gedrag van de testen te vereenvoudigen zijn:

1. *Het verbergen van een aantal visuele elementen uit de test zelf.* Een aantal elementen zijn verborgen door het toevoegen of overschrijven van het `display`-attribuut met de waarde `none` via de `.scss`-bestanden in het thema[61]. Zo wordt bijvoorbeeld de functionaliteit om vragen te markeren (een soort van bladwijzer) verborgen, omdat leerlingen de les en de activiteiten van voor naar achter doorlopen. Ook de voortdurende melding welke beoordelingsmethode er wordt gebruikt en het (voorlopig) toegekende cijfer wordt verborgen. Die functionaliteit is namelijk ontwikkeld voor scholen die hun examens of testen willen organiseren via Moodle. Op Co-De gaat het meer om inhoudsgerichte vragen, waar de leerkracht al dan niet zelf nog evaluatiecriteria aan kan verbinden.

2. *Het verbergen van de overgangs- en bevestigingspagina's.* Enkele functies uit de `renderer`[62] van de test-activiteit zijn overschreven vanuit de `core_renderer`[63] in het eigen thema voor Co-De[64] om overbodige tussenpagina's weg te werken.

   Door het aanpassen van de functie `summary_page` worden een aantal pagina's na het beantwoorden van de laatste vraag en voor het krijgen van de feedback weggewerkt. Op Co-De zal men immers steeds kunnen teruggaan en de inzending bewerken. Herhaaldelijk navragen of men zeker is en het antwoord wenst te verzenden is dus overbodig en storend.

[58] https://docs.moodle.org/34/en/Questions.
[59] https://moodle.org/plugins/browse.php?list=category&id=29.
[60] https://docs.moodle.org/34/en/Quiz_settings.
[61] $code/resources/theme_code.zip.
[62] **bestand**: moodle\mod\quiz\renderer.php.
[63] **bestand**: moodle\theme\code\classes\core_renderer.php.
[64] $code/resources/theme_code.zip.

Een test moet telkens expliciet “geopend” worden (binnen het venster van de les) via een afzonderlijke knop op de portaalpagina van de test. Dit is overbodig op Co-De, omdat leerlingen onmiddellijk de vragen moeten kunnen zien en beantwoorden. Via de functie `view_page` wordt de portaalpagina, die steeds de test inleidt, overgeslagen. Deze wordt wel geladen, maar men wordt meteen doorgestuurd naar de start van de test.

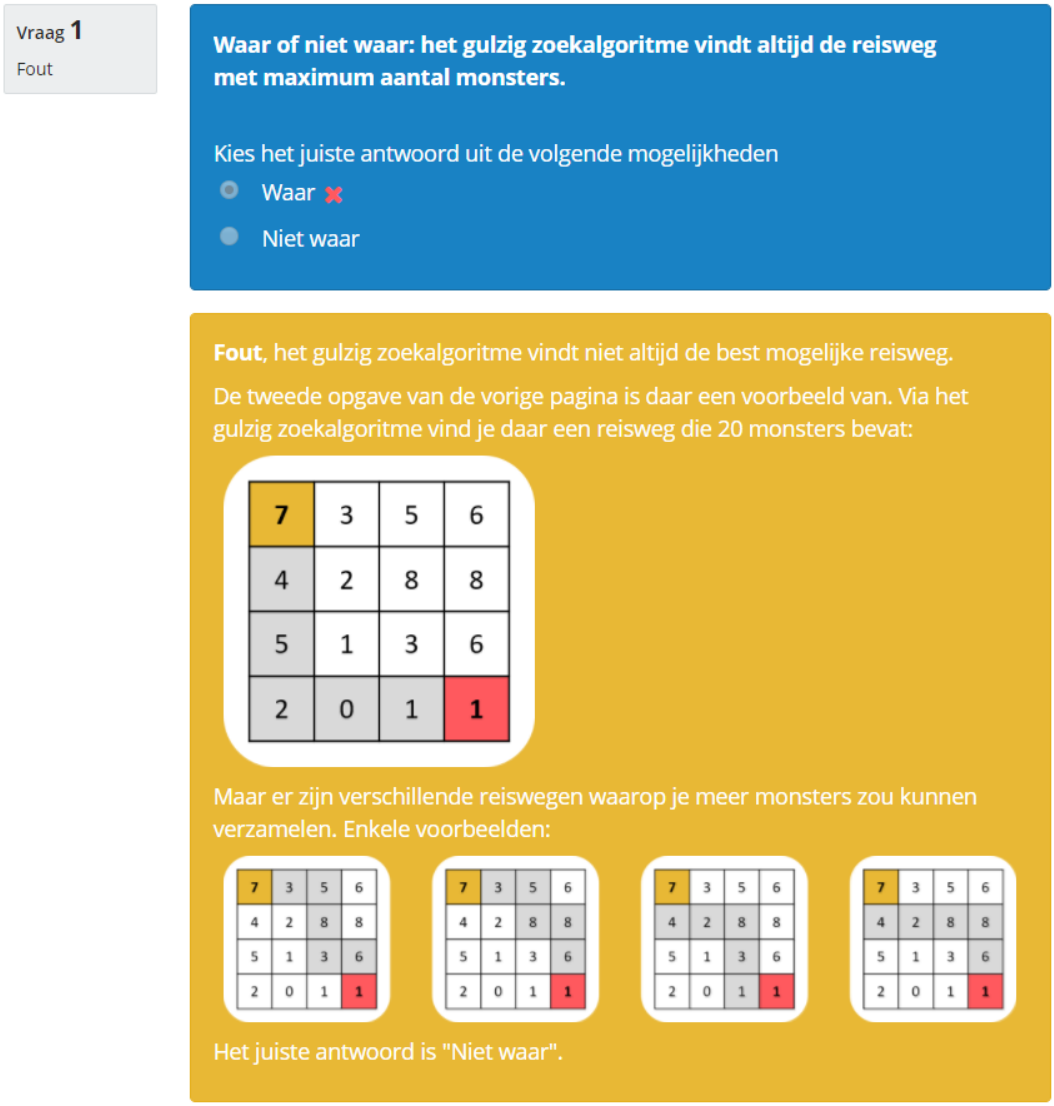
Binnen de functie `review_page` is het overzicht van de feedback weggelaten aangezien er reeds genoeg manieren zijn om feedback te geven.

3. *Toegang tot de volgende activiteit.* Door een combinatie van de twee bovenstaande ingrepen wordt de link naar de volgende activiteit verborgen tot men de laatste vraag heeft ingevuld (zie ook 3.5.1.2). Detecteren wanneer dit het geval is, gebeurt in de functie `attempt_page` van de `renderer`. Het verbergen zelf gebeurt via de `.scss`-bestanden.

4. *Het kleurenschema.* De standaardkleuren voor juiste, gedeeltelijk juiste en foute antwoorden en voor feedback zijn aangepast aan het kleurenschema van Co-De (zie figuren 3.11 en 3.18). Dit gebeurde eveneens via de `.scss`-bestanden in het thema.

De technische appendix (A) geeft verdere toelichting bij al deze wijzigingen. De meeste van de hierboven beschreven aanpassingen situeren zich in het eigen thema voor Co-De (A.8).

Enkele voorbeelden van testen zijn terug te vinden in figuren 3.17 en 3.18 of via deze snelkoppelingen:
- `$code/mod/quiz/view.php?id=31`
- `$code/mod/quiz/view.php?id=154`
- `$code/mod/quiz/view.php?id=42`



Figuur 3.18: Een voorbeeld van feedback op een juist-fout vraag uit de marslander waarbij de leerling het foute antwoord gaf. De foute keuze van de leerling wordt toegelicht aan de hand van enkele tegenvoorbeelden.

3.5.2.6 Programmeeropdracht

De programmeeropdracht heeft de vorm van een vraagtype voor de test-activiteit (zie 3.5.2.5). Bijgevolg kan een programmeeropdracht voor, tussen of na andere vragen in een test worden geplaatst. De programmeeropdracht heeft dezelfde evaluatiemogelijkheden als andere vraagtypes. Zoals uitgelegd in sectie 3.4.3.1 (p. 33) worden de programmeeropdrachten mogelijk gemaakt door de plugin "CodeRunner".

De programmeeropdracht (figuur 3.19) is visueel opgebouwd met de volgende onderdelen: (a) algemene uitleg over de opdracht, (b) een link naar de downloadbare opgavebestanden of onderliggende gebruikte bestanden, (c) een aantal test cases die de implementatie testen, (d) een editor voor de implementatie, (e) twee mogelijkheden om de testen uit te voeren ("precheck" en "controleer") en (f) gepaste feedback over de implementatie van de leerling.

CodeRunner biedt configuratiemogelijkheden om het gedrag van de opdrachten aan te passen. Belangrijk hierbij is het verschil tussen de uitvoermodi "precheck" en "controleer". "Precheck" zal de implementatie van de leerling enkel nakijken op de afgebeelde testgevallen (onderdeel (c) op figuur 3.19). "Controleer" daarentegen evalueert de volledige opdracht en test mogelijk nog op onzichtbare gevallen die de lesontwikkelaar of leerkracht heeft ingegeven (te zien in de feedback, onderdeel (f) op figuur 3.19).

De standaard programmeertaal voor de opdrachten in de *demo-lessen* is Python3. Python is een laagdrempelige programmeertaal die al een vaste plaats heeft binnen het onderwijs [30]. De programmeeroefeningen in Co-De zijn bedoeld voor leerlingen die al programmeerervaring hebben. CodeRunner ondersteunt nog een reeks andere programmeertalen: Python2, C, C++, Java, PHP, JavaScript en Octave. De leerkracht kan ervoor kiezen om in zijn of haar *eigen lessen* de programmeertaal aan te passen.

Enkele voorbeelden van programmeeropdrachten zijn te vinden via onderstaande snelkoppelingen en in figuur 3.19:
- `$code/mod/quiz/view.php?id=156`
- `$code/mod/quiz/view.php?id=188`
- `$code/mod/quiz/view.php?id=171`

Figuur 3.19: Een voorbeeld van een programmeeropdracht in een test-activiteit waarop de verschillende onderdelen staan aangeduid.

3.5.2.7 Feedback

De feedback-activiteit[65] in Moodle sluit nauw aan bij de onderzoek-activiteit[66] (*survey*). Beide zijn erop gericht om de mening van de deelnemer te vragen aan de hand van een vragenlijst. Anders dan bij testen zijn antwoorden hier nooit juist of fout en kan er geen feedback of quotering worden toegekend door de leerkracht. Het verschil tussen een onderzoek- en een feedback-activiteit is dat binnen een onderzoek enkel kan worden gekozen voor een standaard vragenlijst met vragen over de les en de begeleiding. Deze vragenlijsten zijn slechts beperkt aanpasbaar voor eigen gebruik. Feedback-activiteiten zijn daarentegen zeer aanpasbaar, gelijkaardig aan een test (zie 3.5.2.5), maar dan met een ander doel.

In Co-De wordt op het einde van iedere les expliciet ingegaan op de elementen van computationeel denken (zie figuur 3.20) die daar aan bod kwamen. Het feedbackformulier hiervoor is opgedeeld per element van computationeel denken. Bij elk element wordt eerst bevraagd waar en hoe dat element volgens de leerling aan bod kwam. Daarna wordt toegelicht in welke delen dat element volgens de lesontwerpers aan bod kwam, maar uiteraard kan een leerling aangeven dat hij nog op andere manieren aan CD heeft gedaan. Meer informatie over deze vragenlijst is te vinden in sectie 3.5.1.1 en hoofdstuk 4). Van dit feedbackformulier wordt ook een sjabloon aangeboden[67] dat kan gebruikt worden voor nieuwe lessen.

Figuur 3.20: De vijf elementen van CD en hun symbolen.

Buiten het verbergen van enkele overbodige elementen, op dezelfde manier als bij testen (zie 3.5.2.5), is voor de feedback-activiteit niets aangepast voor Co-De. De `html`-ondersteuning (zie 3.5.2.1) wordt wel ten volle benut om een aantrekkelijk en herkenbaar formulier te creëren.

Enkele voorbeelden van de feedbackformulieren rond computationeel denken zijn terug te vinden via deze snelkoppelingen of in figuur 3.6:
- `$code/mod/quiz/view.php?id=21`
- `$code/mod/quiz/view.php?id=197`
- `$code/mod/quiz/view.php?id=54`

[65] https://docs.moodle.org/34/en/Feedback_activity.
[66] https://docs.moodle.org/34/en/Survey_activity.
[67] https://docs.moodle.org/34/en/Feedback_templates.

3.5.3 Rollen

Binnen het leerplatform Co-De is het noodzakelijk om verschillende rechten toe te kennen aan verschillende soorten gebruikers. Leerkrachten moeten de lessen kunnen configureren en de gegevens van de leerlingen raadplegen en aanpassen. Ze moeten ook kunnen zoeken naar geschikte lessen op het platform en die lessen eens kunnen uitproberen. Leerlingen zouden daarentegen enkel hun eigen gegevens mogen zien en de voor hen beschikbare lessen volgen. Alle lesonderdelen mogen meteen zichtbaar zijn wanneer de leerkracht de les bekijkt. De leerling mag de les doorgaans enkel stap-voor-stap doorlopen. Voor de *demo-lessen* ontstaat een extra nood voor een nieuwe soort gebruiker; een demo-leerkracht (zie 3.5.3.2).

Die complexe vereiste is een van de hoofdredenen voor de keuze voor Moodle in plaats van het opbouwen van een eigen platform (zie 3.3.1). Moodle werkt met rollen voor gebruikers[68]: aan deze rollen zijn bepaalde rechten en beperkingen verbonden. Via toelatingen (*permissions*[69]) voor bepaalde aspecten en operaties worden de rollen opgesteld. De mogelijkheden zijn hierbij zeer uitgebreid aangezien de lijst met verschillende aspecten waartoe toegang kan worden verleend zeer ruim is. Gelukkig zijn in Moodle al enkele standaard rollen ingebouwd[70] zoals leerlingen, leerkrachten met of zonder bewerkrechten, beheerders en gasten. Men kan die rollen eenvoudig aanpassen[71] of ze gebruiken als archetype voor nieuwe rollen[72].

Geregistreerde gebruikers hebben in Moodle geen vaste rol doorheen het hele platform, op enkele uitzonderingen na. Rollen worden immers meestal toegewezen per les[73]. Een gebruiker kan dus in principe een leerkracht zijn van een bepaalde les en binnen een andere les als leerling ingeschreven zijn. Een les kan ook meerdere leerkrachten hebben en gebruikers kunnen binnen één les ook meerdere rollen hebben.

In onderdelen 3.5.3.1 tot en met 3.5.3.4 worden de meest voorkomende rollen in Co-De verder toegelicht. Voor de *demo-lessen* (zie 3.5.1.3) geldt dat men zichzelf kan inschrijven als leerling of demo-leerkracht. Bij *eigen lessen* (zie 3.5.1.3) zijn ook de andere rollen mogelijk. Een overzicht van de verschillende rollen en hun rechten is terug te vinden in tabel 3.2.

[68] https://docs.moodle.org/34/en/Roles_and_permissions.
[69] https://docs.moodle.org/34/en/Permissions.
[70] https://docs.moodle.org/34/en/Standard_roles.
[71] https://docs.moodle.org/34/en/Managing_roles.
[72] https://docs.moodle.org/34/en/Creating_custom_roles.
[73] https://docs.moodle.org/34/en/Assign_roles.

	Rol				
	Leerling	Demo-leerkracht	Leerkracht zonder bewerkrechten	Leerkracht met bewerkrechten	Cursusbeheerder
Rol mogelijk in een *demo-les*	+	+			+
Zelfregistratie mogelijk bij een *demo-les*	+	+			
Rol mogelijk in een *eigen les*	+		+	+	+
Rechten					
Kan de les volgen	+	+	+	+	+
Moet de les stap-voor-stap doorlopen	+				
Kan verborgen delen zien		+	+	+	+
Kan antwoorden zien			+	+	+
Kan antwoorden beoordelen			+	+	+
Kan de les aanpassen				+	+
Kan leerlingen toevoegen				+	+
Kan leerkrachten toevoegen					+
Kan lessen toevoegen of verwijderen					+

Tabel 3.2: Het overzicht van welke rollen mogelijk zijn bij welke soort les en van de rechten per rol.

3.5.3.1 Leerling

De leerlingenrol binnen Co-De is de rol met de minste rechten. Leerlingen hebben geen toegang tot de instellingen van de lessen en lesonderdelen. Ze kunnen ook niet zien welke andere gebruikers verbonden zijn aan de les en welke rol zij hebben. Verder dient een leerling steeds het sequentiële verloop van de les te volgen. Activiteiten worden pas zichtbaar en/of beschikbaar op het moment dat de voorwaarden hiervoor voldaan zijn (zie 3.5.1.2). Leerlingen kunnen nooit lesonderdelen zien die verborgen zijn, zoals de informatie gericht aan de leerkrachten (zie 3.5.1.1). Alle antwoorden van de leerlingen worden bijgehouden. Tot slot kunnen leerlingen worden onderverdeeld in groepen en afhankelijk van hun groep(en) toegang krijgen tot bepaalde lesonderdelen[74].

Aan de rechten van deze rol is niets gewijzigd ten opzichte van de standaard Moodle leerling-rol[75].

[74] https://docs.moodle.org/34/en/Groups.
[75] https://docs.moodle.org/34/en/Student_role.

3.5.3.2 Demo-leerkracht

Deze rol is niet standaard aanwezig in Moodle. Hij werd toegevoegd aan Co-De om de *demo-lessen* te bekijken vanuit het perspectief van de leerkracht (zie 3.5.1.3). Demo-leerkrachten hebben meer rechten dan de leerlingen (zie 3.5.3.1) en minder dan leerkrachten (zie 3.5.3.3). Een demo-leerkracht kan, in tegenstelling tot een leerling, wel de verborgen delen van lessen zien en al de hele les bekijken zonder ze te moeten volgen. Verder kan een demo-leerkracht, in tegenstelling tot een leerkracht, de lessen niet aanpassen en geen persoonsgegevens of antwoorden raadplegen van leerlingen (zie 3.5.1.3).

Aangezien deze rol slechts enkele toelatingen meer heeft dan de leerlingenrol, en veel minder dan de leerkrachtenrol werd deze opgesteld vanuit het archetype leerling[76]. Vertrekkend van de leerlingenrol zijn rechten om verborgen delen te zien verleend aan deze nieuwe rol.

3.5.3.3 Leerkracht

Voor leerkrachten zijn er twee verschillende rollen mogelijk: leerkrachten met bewerkrechten (*teacher*[77]) en leerkrachten zonder bewerkrechten (*non-editing teacher*[78]). Deze rollen zijn binnen Co-De integraal behouden zoals ze worden aangeboden door Moodle. Onder "bewerkrechten" vallen de rechten om een les en lesonderdelen aan te passen, van plaats te wisselen, toe te voegen en te verwijderen. Leerkrachten zonder bewerkrechten hebben hiervoor geen rechten, maar ze kunnen wel de gegevens en antwoorden van hun leerlingen zien en beoordelen.

Wellicht ontstaat binnen Co-De in de toekomst de vraag naar leerkrachtenrollen met toegangsrechten die tussen de twee standaard soorten in liggen. Zo zouden leerkrachten bijvoorbeeld net voldoende rechten kunnen krijgen om het leerpad te configureren, zonder zich zorgen te moeten maken om fouten te maken. De exacte invulling van deze nieuwe rol(len) moet echter worden afgetoetst met echte gebruikers die al vertrouwd zijn met Co-De. Meer informatie over Co-De in de toekomst kan men terugvinden in hoofdstuk 5.

[76] https://docs.moodle.org/34/en/Creating_custom_roles#Role_archetypes.
[77] https://docs.moodle.org/34/en/Teacher_role.
[78] https://docs.moodle.org/34/en/Non-editing_teacher_role.

3.5.3.4 Beheerder

Ook de beheerdersrol heeft twee varianten: cursusbeheerder en sitebeheerder.

Cursusbeheerder
Een cursusbeheerder[79] (of lesbeheerder) is een rol die kan worden toegekend per les zoals de voorgaande rollen. Een beheerder heeft alle rechten van een leerkracht met bewerkingsrechten (zie 3.5.3.3). Dit houdt onder andere in dat de cursusbeheerder de antwoorden van leerlingen uit de les kan raadplegen. Daarenboven kan een beheerder de andere rollen binnen de les toekennen of afnemen. De cursusbeheerder heeft steeds toegang tot alle opties en instellingen van een les en wordt geacht om zowel de les zelf als de andere rollen te beheren.

Aan de rechten van deze rol is niets gewijzigd voor Co-De ten opzichte van de standaard Moodle-rol[80].

Sitebeheerder
Sitebeheerders zijn gebruikers die zijn aangeduid als beheerder van het hele leerplatform. Deze rol is een site-brede rol en ze is niet afhankelijk van les tot les. Een sitebeheerder is per definitie ook cursusbeheerder van alle lessen op het leerplatform. Het is de taak van de sitebeheerder(s) om de Moodle omgeving van binnenuit te beheren. Daaronder valt het aanmaken van lessen, installeren van plugins, beheren van andere gebruikers, aan- en uitzetten van opties, beheren van de taalpakketten en nog heel wat meer. Verschillende gebruikers kunnen de rol van sitebeheerder opnemen.

Aan de rechten van deze rol kan niets worden gewijzigd ten opzichte van de standaard Moodle sitebeheerders-rol[81]. In Co-De is er een *Admin User* aangeduid als hoofd-sitebeheerder bij het opzetten van de omgeving. Daarnaast hebben de twee auteurs van deze masterproef de rol van sitebeheerder. In de toekomst kan deze rol ook aan andere gebruikers worden toegekend om Co-De mee te beheren. Meer info daarover volgt in hoofdstuk 5.

Noot: sitebeheerders staan in voor het beheer van Co-De **binnen** de Moodle-omgeving. Hieronder valt niet het beheren van het deployment van Co-De of van de hardware waarop het leerplatform draait. Het beheer van de Moodle-onderhoudstaken, back-ups en hulpbronnen (zie 3.4) valt hier evenmin onder.

[79] https://docs.moodle.org/34/en/Manager_role.
[80] https://docs.moodle.org/34/en/Manager_role.
[81] https://docs.moodle.org/34/en/Site_administrators.

3.6 Gebruik

Hieronder volgt een beknopt overzicht van hoe men Co-De online kan gebruiken. Daarnaast wordt wat meer uitleg gegeven over de mogelijkheden om op een eigen Moodle-site aan de slag te gaan met de toegevoegde functionaliteiten en lesinhoud.

3.6.1 Gebruikershandleiding

Alle mogelijkheden voor de gebruikers op Co-De worden in detail besproken in de gebruikershandleiding die te vinden is op het platform zelf via `$code/handleiding`. Deze handleiding is toegankelijk voor elke geregistreerde gebruiker en wordt bijgewerkt indien zaken niet duidelijk blijken te zijn of zouden veranderen. De inhoudsopgave van de handleiding wordt weergegeven in tabel 3.3 waarin de koppelingen verwijzen naar de desbetreffende artikelen uit de handleiding.

In de handleiding wordt elke functionaliteit toegelicht aan de hand van een duidelijke beschrijving, begeleid door schermafbeeldingen waarop elementen zijn aangeduid. Een voorbeeld van een pagina uit de handleiding staat afgebeeld in figuur 3.21. Dit artikel beschrijft hoe men als leerkracht de voortgang van de leerlingen kan opvolgen. De inhoud van de handleiding wordt hier niet gekopieerd, aangezien ze op die manier haar dynamisch karakter zou verliezen en ze teveel zou overlappen met de rest van deze tekst.

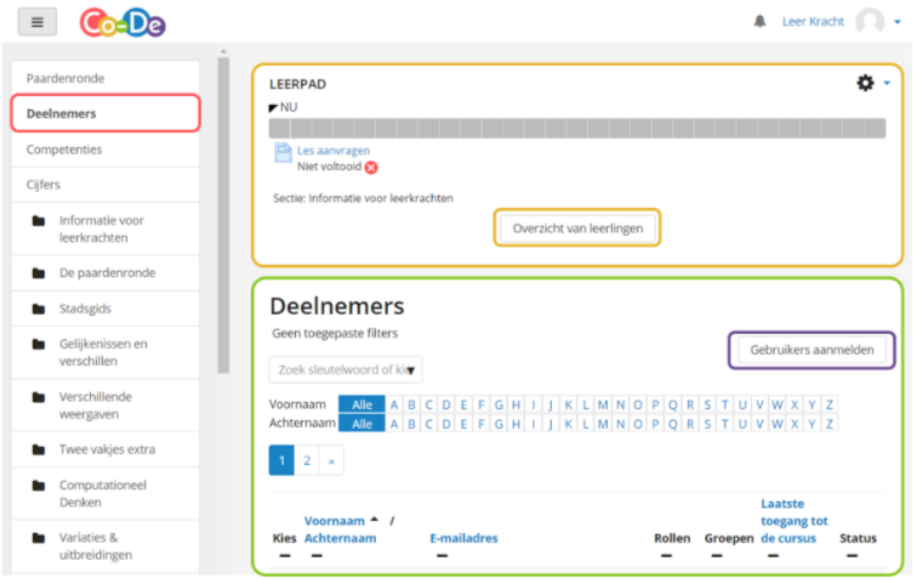
Algemeen		Introductietekst
		Co-De community forum
Wegwijs op het platform		Registreren en inloggen
		Navigatie
		Lessen en categorieën
		Rollen
Computationeel denken		Wat is computationeel denken?
		Waarom computationeel denken?
		Vijf kernelementen
Lessen		Lesstructuur
		Soorten lessen
		Lessen aanvragen
		Lessen toevoegen
		Lessen op Co-De
Een les gebruiken	*Demo-lessen*	Lessen volgen als leerling
		Lessen bekijken als demo-leerkracht
		Zelf aanmelden
	Eigen lessen	Lessen aanvragen
		Leerlingen beheren
		Lessen volgen als leerling
		Les aanpassen
		Antwoorden van leerlingen bekijken
	Op eigen Moodle-site	Lessen
		Plugins
Een les aanpassen		Algemeen
		Activiteiten en secties bewerken
		Activiteiten en secties verbergen
		Activiteiten en secties herordenen
		Activiteiten toevoegen
		Secties toevoegen
		Activiteiten en secties verwijderen
Activiteiten		Algemeen
		Pagina
		- Animatie
		Keuze
		Opdracht
		Test
		- Programmeeropdracht
		Feedback
		- Vragenlijst computationeel denken
Over Co-De		Het project
		Technische aspecten
Contact		E-mailadres
		Met dank aan

Tabel 3.3: De inhoudsopgave van de handleiding (`$code/handleiding`) met snelkoppelingen naar de verschillende delen.

Leerlingen beheren

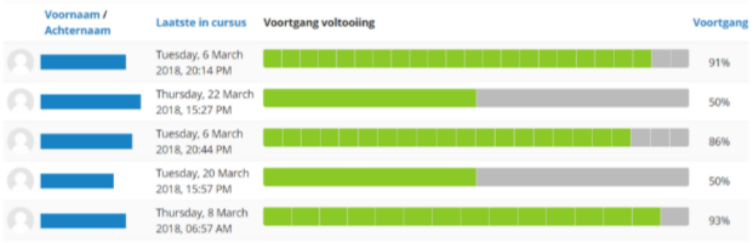
Zowel als *leerkracht met bewerkrechten* als *leerkracht zonder bewerkrechten* kan je je leerlingen beheren in jouw lessen. Je kan dit doen door in het menu aan de linker kant op "**deelnemers**" te klikken (zie schermafbeelding).

Je belandt dan op een pagina met twee grote delen. Een eerste deel waar je de **voortgang van de leerlingen** kan opvolgen en een tweede deel waar je de **aangemelde leerlingen** kan beheren. In dat tweede deel bevindt zich ook een knop om **leerlingen toe te voegen** aan de les.



Voortgang van de leerlingen opvolgen

Wanneer je op de knop "**overzicht van leerlingen**" klikt, die onderaan in de voortgangsbalk voor leerkrachten staan, dan beland je op een scherm dat je toont waar de verschillende leerlingen zich bevinden in de les. Een voorbeeld:



Bij elke leerling wordt vermeld hoe ver hij of zij gevorderd is binnen de delen die hij of zij kan zien. Het aantal vakjes toont hoeveel van de les deze leerling reeds beschikbaar heeft. De vakjes die groen zijn gekleurd zijn activiteiten die deze leerling volledig heeft afgewerkt.

Figuur 3.21: Een deel van het artikel uit de handleiding dat beschrijft hoe een leerkracht de gegevens van de leerlingen kan beheren. Het volledige artikel kan men raadplegen via `$code/mod/page/view.php?id=248`.

3.6.2 Gebruik op een eigen Moodle-site

Naast het gebruik van Co-De via `$code`, zoals het wordt beschreven in de handleiding, is het ook mogelijk om de inhoud en functionaliteiten van Co-De te gebruiken op een eigen Moodle-site. In sommige scholen is Moodle namelijk reeds aanwezig als online platform voor het delen van bestanden, bijhouden van een schoolagenda en dergelijke meer (zie 3.3.2 op p. 29).

De lessen op Co-De maken gebruik van enkele plugins. Sommige plugins moeten verplicht worden geïnstalleerd om de lessen te kunnen gebruiken. Andere zijn eerder optioneel en kan men achterwege laten indien die functionaliteit niet gewenst is voor de eigen Moodle-site.

3.6.2.1 Lessen

De lessen die beschikbaar zijn op Co-De (zie hoofdstuk 4) kunnen via de gebruikelijke procedure hiervoor worden ingeladen op eigen Moodle-sites[82]. De enige voorwaarde hiervoor is dat de verplichte plugins zijn geïnstalleerd (zie 3.6.2.2). Zowel het inladen van lesinhoud als het installeren van plugins zijn acties die zijn voorbehouden aan beheerders van de Moodle-site in kwestie en kunnen niet vanuit Co-De worden uitgevoerd.

Concrete richtlijnen hierover worden beschreven in de gebruikershandleiding van Co-De in dit artikel: `$code/mod/page/view.php?id=251`.

3.6.2.2 Plugins

Alle toegevoegde functionaliteiten beschreven in onderdeel 3.5 worden aangeboden in de vorm van Moodle-plugins. Op die manier kunnen ze eenvoudig worden geïnstalleerd op compatibele Moodle-sites door de beheerder van die site[83]. De zelf ontwikkelde plugins zijn nog niet beschikbaar in de Moodle-bibliotheek met plugins[84], aangezien ze hiervoor manueel moeten worden nagekeken door een Moodle-ontwikkelaar en dit een tijdrovend proces is. De plugins kunnen wel worden gedownload als `.zip`-bestand. Dat bestand kan daarna worden ingeladen via de gebruikelijke procedure[85] op een eigen Moodle-site. Een overzicht van de belangrijkste plugins wordt weergegeven in tabel 3.4 en een uitgebreidere bespreking volgt hieronder.

[82] https://docs.moodle.org/34/en/Activity_restore.
[83] https://docs.moodle.org/34/en/Installing_plugins.
[84] https://moodle.org/plugins/.
[85] https://docs.moodle.org/34/en/Installing_plugins#Installing_via_uploaded_ZIP_file.

	Relatieve voltooiing	CodeRunner	Thema "code"	Voortgangsbalk
Verplicht te installeren	ja	ja[1]	nee	nee
Ontwikkeld voor Co-De	ja	nee	ja	nee
Aangepast voor Co-De	ja	nee	ja	ja
Installeren via `.zip`-bestand mogelijk	ja	ja	ja	ja
Installeren via Moodle-plugin-bibliotheek mogelijk	nee	ja	nee	ja[2]

[1]enkel verplicht voor lessen met programmeeropdrachten.
[2]enkel de basisversie zonder aanpassingen voor Co-De.

Tabel 3.4: Een overzicht van de belangrijkste plugins op Co-De.

Relatieve voltooiing
Deze plugin is volledig ontwikkeld binnen het project van deze masterproef. Hij voegt een extra toegangsbeperking toe bij secties en activiteiten[86]. Via deze plugin kan worden aangegeven dat een bepaalde activiteit of sectie pas zichtbaar/beschikbaar mag worden wanneer de vorige activiteit/sectie volledig is afgewerkt. De werking van deze plugin wordt meer in detail besproken in onderdeel 3.5.1.2. Deze toevoeging zorgt ervoor dat de lesonderdelen zonder zorgen kunnen herordend worden en het lesverloop daarbij gewaarborgd blijft. De implementatie van deze plugin wordt uitgebreid besproken in de technische appendix (zie A.6).

Om de lessen van Co-De te gebruiken op een eigen Moodle-site is deze plugin verplicht. Men kan de plugin downloaden via onderstaande link en hem installeren op Moodle-sites (versie 3.4+) via de gebruikelijke procedure:
`$code/resources/relative_completion.zip`

CodeRunner
Via deze plugin worden de programmeeropdrachten mogelijk gemaakt als vraagsoort bij de testen (zie 3.5.2.6). Deze plugin is verplicht voor lessen met programmeeropdrachten. Voor deze plugin moet ook een Jobe server worden ingesteld zoals reeds werd beschreven bij de technische aspecten van Co-De in 3.4.3.1.

Deze plugin is een standaard plugin die men kan installeren uit de plugin-bibliotheek[87]:
- `https://moodle.org/plugins/qtype_coderunner`
- `http://coderunner.org.nz/`

[86]`https://docs.moodle.org/34/en/Restrict_access_settings`.
[87]`https://moodle.org/plugins/`.

Thema "code"
Co-De heeft zijn eigen lay-out en gebruikersinterface. Deze zijn mogelijk gemaakt via een eigen *child-theme* van het standaard thema "Boost"[88]. In dit thema, genaamd "code", worden heel wat kleine wijzigingen aan het gedrag en de weergave van Moodle gebundeld. Veel van deze aanpassingen werden reeds beschreven in de bespreking van het ontwerp en de implementatie van Co-De (zie 3.5). Naast deze aanpassing wordt via dit thema ook een uniforme en herkenbare stijl en kleurengebruik voorzien op Co-De.

Deze plugin is niet verplicht om met de lessen van Co-De te kunnen werken op een eigen Moodle-site. Hij is integraal ontwikkeld in het kader van deze masterproef en hij is beschikbaar via onderstaande snelkoppeling. Er wordt in deze tekst ook herhaaldelijk naar deze snelkoppeling verwezen in de voetnoten van andere delen.
`$code/resources/theme_code.zip`

De implementatie van deze plugin wordt uitgebreid besproken in de technische appendix (zie A.8).

Voortgangsbalk
Bovenaan elke les staat een voortgangsbalk die aangeeft welke activiteiten al voltooid zijn en welke nog aan bod moeten komen (zie 3.5.1.2). De plugin die dit mogelijk maakt, is gebaseerd op de standaard Moodle-plugin "Completion Progress" uit de plugin-bibliotheek[89]. Om de voortgangsbalk bovenaan de les te kunnen plaatsen (en niet uitsluitend in een zijbalk), is de originele plugin aangepast en werden enkele functies toegevoegd aan het thema voor Co-De. De aangepaste plugin zal dus enkel werken in combinatie met het standaard thema Boost of het eigen thema code (zie hierboven).

Deze plugin is niet verplicht om met de lessen van Co-De te kunnen werken op een eigen Moodle-site. Hij is aangepast door de ontwikkelaars van Co-De en beschikbaar via: `$code/resources/progress_bar.zip`.

De implementatie van deze plugin wordt uitgebreid besproken in de technische appendix (zie A.7).

[88] https://docs.moodle.org/34/en/Boost_theme.
[89] https://moodle.org/plugins/block_completion_progress.

3.7 Testen

Doorheen het academiejaar 2017-2018 werden heel wat verschillende aspecten van Co-De getest door verschillende personen. Deze testen hebben een belangrijke invloed gehad op bepaalde ontwerpkeuzes voor het leerplatform. Hieronder volgt een overzicht van de belangrijkste testen. Verder is ook elk aspect van het leerplatform uitvoerig getest tijdens de ontwikkeling in kleinere, losstaande testen. Hiervoor werden op Co-De een vijftal *dummy* leerling-accounts en een *dummy* leerkracht-account aangemaakt en werd herhaaldelijk een beroep gedaan op de begeleiders van deze masterproef.

3.7.1 Gebruikerstest

Op 6 maart werd het platform onderworpen aan een eerste gebruikerstest door een groep vrijwilligers die simultaan inlogden en een les volgden. Een twintigtal testpersonen baande zich een weg op het platform en volgde zelfstandig de les paardenronde & stadsgids. De belangrijkste resultaten van deze test zijn samengevat in appendix B.

Uit deze test bleek dat de servercapaciteit en reactietijd van het netwerk voldoen aan de eisen. Deze test simuleerde immers een realistische belasting van één klas waarvan alle leerlingen simultaan werken op Co-De.

Daarnaast werden door deze test enkele pijnpunten blootgelegd die later verholpen werden. Zo was er een tijdelijke storing bij de mailserver die verantwoordelijk was voor het versturen van de aanmeldlink naar nieuwe gebruikers (zie "Verliep het inloggen vlot?" in tabel B.1). Daarom werd de functionaliteit toegevoegd om gebruikers zichzelf te laten registreren op het platform en werd de mogelijkheid om zichzelf aan te melden voor *demo-lessen* ingevoegd. Daarnaast bleek de (toen nog complexe) navigatie op het platform een werkpunt. Deze navigatie werd dan ook grondig omgevormd zoals reeds werd beschreven in 3.5.1.2.

De meeste testpersonen, die voordien nog geen ervaring hadden met Co-De of CD, gaven aan dat ze iets hadden bijgeleerd en dat ze het leuk vonden om de les te volgen (zie B.1). Ook de antwoorden die ze gaven op de vragen binnen de les hebben ons geholpen om deze les en het platform in het algemeen verder uit te bouwen (zie B.2).

Tot slot bleek ook uit deze test dat de vragenlijst over computationeel denken, achteraan in de les, de testpersonen iets heeft bijgebracht. Het expliciteren van de elementen van CD en waar deze aan bod kwamen, heeft hen aangezet tot nadenken en doen inzien dat ze de verworven kennis ook kunnen toepassen in andere situaties. De antwoorden op deze feedback-activiteit worden weergegeven in B.2.3. Hieruit blijkt dat de testpersonen vrij goed kunnen identificeren welke elementen van CD ze op welke plaats gebruikt hebben, aangezien hun antwoorden een grote overeenstemming vertonen met wat wordt aangegeven in 4.3.3. Ook de voorbeelden die worden gegeven bij de vraag "Kan je nog andere problemen bedenken die je zou kunnen omvormen tot een route vinden in een graaf?" illustreren dat de testpersonen iets geleerd hebben en dat ze de geleerde kennis denken te kunnen gebruiken in andere situaties.

3.7.2 Losse testen

Naast de georganiseerde gebruikerstest op 6 maart zijn de verschillende componenten van Co-De getest doorheen de ontwikkeling. Hieronder volgen enkele bevindingen die uit zulke testen voortkwamen en die de verdere ontwikkeling van Co-De hebben beïnvloed.

Bij de eerste testen met afgewerkte lessen bleek dat de test-activiteit nogal omslachtig was om mee te werken als leerling. De vele overbodige tussenpagina's en ongebruikte functionaliteiten werkten storend voor heel wat testgebruikers. Deze activiteit werd dan ook grondig herwerkt voor Co-De zoals werd beschreven in onderdeel 3.5.2.5. Hetzelfde gold, in mindere mate, voor de opdracht- en feedback-activiteiten.

De interactieve animaties (zie 3.5.2.2) die worden ingesloten op pagina's bleken niet op elke webbrowser en besturingssysteem te werken. De locatie van de bronbestanden bleek hiervan de oorzaak te zijn, in combinatie met het gebruik van `HTTP` in plaats van `HTTPS`. De oplossingen voor deze problemen werden reeds beschreven in respectievelijk 3.4.3.2 en 3.4.2.

Vele andere aanpassingen die reeds werden beschreven bij het ontwerp en de implementatie (zie 3.5) kwamen voort uit problemen die ontdekt werden doorheen de vele kleine testen.

3.7.3 Test in klassituatie

Co-De werd ook eenmalig uitgetest door een echte klas[90]. Deze test was vooral nuttig om het aanvraagformulier voor leerkrachten te testen en na te gaan of de leerkracht aan de slag kon met een *eigen les*. Dit verliep zeer vlot en de beheerders van Co-De hebben geen extra ondersteuning moeten bieden aan de leerkracht of de leerlingen. Zij vonden zelf hun weg op het platform en vonden het een leuke en leerzame ervaring.

[90] De klas die deelnam was een zevende jaar stuur- en beveiligingstechnieken uit het Damiaaninstituut Aarschot. De leerlingen volgden zelfstandig, maar in groep, de les paardenronde & stadsgids tijdens de laatste les van hun semester. Ze hadden geen voorbereiding gekregen en maakten gebruik van het standaard leerpad.

3.8 Besluit

Dit hoofdstuk legt, samen met een uitgebreide digitale gebruikershandleiding en een technische appendix, de opbouw en werking van ons leerplatform uit dat online beschikbaar is via https://code.experiments.cs.kuleuven.be/.

Het platform vertrekt van de standaard Moodle-configuratie die op een server opgezet is aan de hand van Docker. Er is gekozen voor Moodle als onderliggend platform omdat hierin reeds vele basisfunctionaliteiten ingebouwd zijn en er mogelijkheden zijn om zaken aan te passen of toe te voegen. Het basispakket is langzaamaan uitgebouwd tot een webapplicatie met een consistent kleurenpatroon en thema, eenduidige manier van navigeren en een vaste lesstructuur.

Op het platform is een onderscheid gecreëerd tussen zogenaamde *demo-lessen* (ter illustratie) en *eigen lessen* (voor een specifieke klas of groep). Binnen deze lessen worden diverse soorten activiteiten ondersteund: pagina's (met tekst, afbeeldingen, video's of animaties), keuzes, opdrachten, testen (met of zonder programmeeropdrachten) en feedbackformulieren (inclusief een vragenlijst over CD). Om de rechten van gebruikers te regulariseren zijn er verschillende rollen op het platform en binnen de lessen: leerling, demo-leerkracht, leerkracht en beheerder.

Zowat alle toegevoegde functionaliteiten worden aangeboden in de vorm van plugins voor Moodle. Hierdoor kunnen ze ook worden toegevoegd aan andere Moodle-sites. Vooral de relatieve voltooiing is, los van Co-De, ook zinvol voor andere Moodle gebruikers.

De verschillende componenten van Co-De zijn onderworpen aan enkele testen en het platform werd reeds gebruikt door een echte klas.

Hoofdstuk 4

Lessen op Co-De

4.1 Inleiding

In dit hoofdstuk wordt dieper ingegaan op de lessen van Co-De. Eerst worden enkele algemeenheden over het ontwerp, de opbouw en de aanpasbaarheid van de lessen besproken. Daarna wordt elke les op Co-De meer in detail besproken en volgt een besluit.

Het eerste deel bestaat uit vier onderdelen. Ten eerste wordt het 4C/ID-model toegelicht dat gebruikt wordt om verschillende ontwerpkeuzes van het platform en de lessen te staven. Daarna wordt toegelicht hoe een les, het leerpad en de lesonderdelen worden ontworpen. Hierbij wordt ook een kader beschreven dat de lesactiviteiten karakteriseert aan de hand van vier begrippen: *doe*, *denk*, *reflecteer* en *illustreer*. Dit kader wordt vervolgens toegepast op de vier uitgewerkte lessen. Ten derde wordt kort het onderscheid tussen de soorten lessen aangehaald en tot slot worden de mogelijkheden om de lessen aan te passen besproken.

	Abstractie	Veralgemening	Decompositie	Algoritmisch Denken	Evaluatie
Paardenronde & stadsgids	+	+			
Doolhoven	+			+	+
De marslander			+	+	+
De drie bekers	+	+			

Tabel 4.1: De verdeling van de kernelementen van computationeel denken over de verschillende lessen op Co-De. Een lege cel betekent dat het kernelement mogelijk wel aanwezig is in die les, maar niet centraal staat.

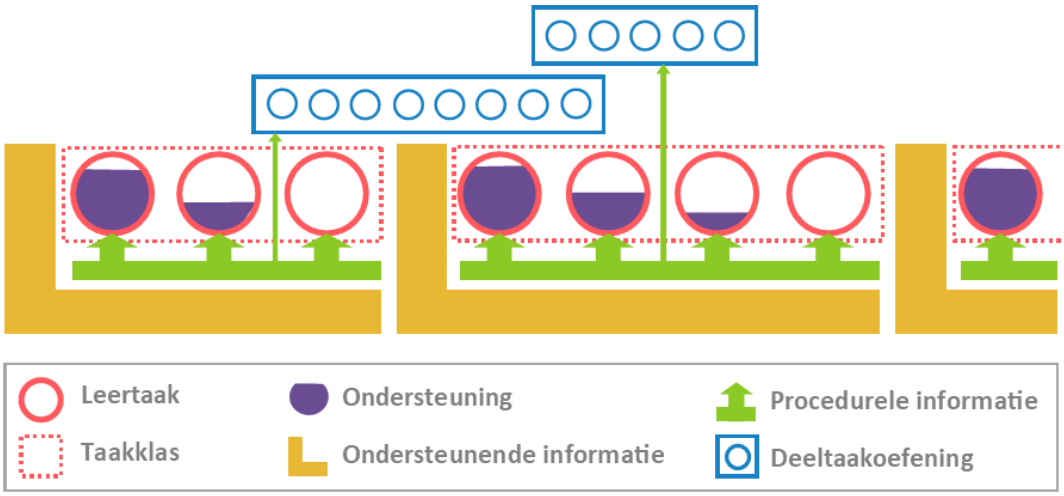
In het tweede gedeelte worden de vier lessen op Co-De meer in detail besproken, in de volgorde waarin ze werden ontwikkeld: paardenronde & stadsgids, doolhoven, de marslander en de drie bekers. Bij elke les wordt eerst de probleemstelling geschetst en het standaard leerpad toegelicht. Vervolgens wordt uiteengezet in welke delen van de les bepaalde elementen van computationeel denken aan bod komen (tabel 4.1). Tot slot wordt ingezoomd op de ontwerpkeuzes die zijn gemaakt bij het bouwen van de les. Daar wordt toegelicht waarom bepaalde activiteitensoorten gekozen zijn, welke alternatieve leerpaden mogelijk zijn en welke compromissen er gemaakt moesten worden bij het uitwerken van de les.

4.2 Algemeen

4.2.1 4C/ID-model

Vaardig worden in computationeel denken is niet eenvoudig en vereist heel wat oefening en herhaling. De vaardigheden die via CD worden aangeleerd zijn niet enkel nuttig binnen informatica. Ook binnen andere kennisdomeinen en in het dagelijks leven komen deze vaardigheden van pas (zie hoofdstuk 2). Om computationeel denken op Co-De op een kwaliteitsvolle en onderbouwde manier aan te leren, is dus ook enige achtergrond op het vlak van instructiemodellen vereist.

Wereldwijd wordt op heel wat plaatsen het 4C/ID-model gebruikt om opleidingen, cursussen en opleidingstrajecten vorm te geven. Ook binnen de KU Leuven wordt dit model aangewend, bijvoorbeeld bij het Centrum voor Blended Learning (CBL) van de Kulak [6]. Het 4C/ID-model dateert van 2002 en staat voor “Four Component Instructional Design Model” of “Vier Componenten Instructie Ontwerp Model” [33]. Het gaat om een instructie-design-model, wat betekent dat er gewerkt wordt rond het aanleren van abstracte concepten door middel van toepassingen of artefacten verbonden met het dagelijkse leven [11]. 4C/ID is een model dat leertrajecten ontwerpt aan de hand van vier componenten [33]: leertaken, ondersteunende informatie, procedurale informatie en deeltaakoefening [6]. Figuur 4.1 geeft die componenten schematisch weer. Van Merriënboer en Kirschener, de bedenkers van het 4C/ID-model, schreven later ook een boek om concreet aan de slag te gaan met het model, getiteld “Ten steps to complex learning: a systematic approach to four-component instructional design” [33]. Hieronder volgt een bondige bespreking van de belangrijkste elementen uit het model. Daarna wordt ingegaan op wat voor Co-De relevant is. In de rest van dit hoofdstuk worden ook regelmatig keuzes gestaafd aan de hand van dit model.



Figuur 4.1: Een schematisch overzicht van de componenten uit het 4C/ID-model. De figuur is geïnspireerd op een figuur uit het boek "Ten Steps to Complex Learning" [33, p. 13].

4.2.1.1 Vier componenten

De leertaken (*learning tasks*) vormen de eerste component uit het model. Deze component wordt afgebeeld door rode cirkels in figuur 4.1. Een leertaak is een oefening of opdracht die liefst zo realistisch mogelijk is. De leertaken worden gegroepeerd in taakklassen van ongeveer dezelfde moeilijkheidsgraad [33]. Binnen de taakklas wordt steeds minder ondersteuning aangeboden, zodat de leerling steeds autonomer gaat werken [6].

De ondersteunende informatie (*supportive information*), die wordt afgebeeld als gele L-vormige blokken in figuur 4.1, vormt de tweede component. Ondersteunende informatie is informatie die beschikbaar is voor de hele taakklas [33]. Ze geeft ondersteuning bij de aspecten die niet geroutineerd moeten worden [6].

Procedurale informatie (*procedural information*) daarentegen wordt gegeven per leertaak en wordt afgebeeld door groene pijlen in figuur 4.1. Deze informatie biedt eveneens ondersteuning bij de aspecten die niet geroutineerd moeten worden, maar dan gefocust op één leertaak in plaats van op de hele taakklas [33].

De vierde component is de deeltaakoefening (*part-task practice*) die wordt afgebeeld in het blauw in figuur 4.1. Deze component is optioneel [33] en heeft als doel om specifieke leertaken te automatiseren door ze herhaaldelijk toe te passen [6].

4.2.1.2 Het 4C/ID-model binnen Co-De

Het doel van het 4C/ID-model is om een kader aan te bieden om trainingsprogramma's te ontwikkelen voor complexe taken [33]. Complexe taken zijn taken

waarin zowel kennis, vaardigheden als attitudes aan bod komen [6]. Dit is ook het geval bij leren computationeel denken (zie hoofdstuk 2). Zowel op onderzoeks- als op praktijkniveau werd het 4C/ID-model reeds onderzocht en gevalideerd [6]. Binnen het domein programmeren werd ook de effectiviteit ervan nagegaan [5, 32, 26]. De *learning gain* (leerwinst) is veel hoger bij leerlingen die onderwezen worden via het 4C/ID-model dan bij leerlingen waarvoor dit niet het geval is [26, p. 216-217]. Bij leertrajecten waar daarenboven gebruik wordt gemaakt van een ICT-toepassing (bijvoorbeeld een online platform) ligt de *learning gain* nog hoger [26, p. 216-217]. Dat maakt het model bijzonder interessant om ook binnen de context van Co-De te gebruiken.

Binnen het 4C/ID-model wordt veel belang gehecht aan de volgorde van de leertaken en de opdeling in taakklassen [33]. Deze opdeling heeft ook impact op wanneer welke ondersteunende en procedurale informatie wordt aangeboden. Een goede volgorde is echter afhankelijk van de leerlingen (lerenden), van hun achtergrond en voorkennis [33]. Vandaar dat op Co-De heel wat flexibiliteit wordt geboden om de leerpaden, secties en activiteiten aan te passen. Een goede volgorde en configuratie van de ondersteunende en procedurale informatie is zelfs verschillend van lerende tot lerende [33]. Geïndividualiseerde configuraties zijn op Co-De niet mogelijk. Desalniettemin biedt een interactief online leerplatform hier heel wat meer mogelijkheden dan een traditioneel handboek of een leerkracht die iets klassikaal uitlegt.

Via een *holistische benadering* probeert het 4C/ID-model tegemoet te komen aan drie veelvoorkomende problemen binnen het educatief domein [33]. Men tracht compartimentalisatie te voorkomen, fragmentatie te verminderen en transfer naar andere leerprocessen en domeinen te bevorderen [5]. Deze aanpak staat haaks op een *atomisch ontwerp* waarin taken worden aangeleerd door kleine delen afzonderlijk aan te leren [33]. Binnen Co-De wordt ook gewerkt met de *holistische benadering* waarvan de effectiviteit reeds bewezen is [31, 32, 33].

Een *holistische benadering* betekent echter niet dat het zinloos is om computationeel denken op te delen in vijf kernelementen en in bepaalde lessen te focussen op bepaalde elementen. Het doel van het leerproces is immers om CD te kunnen gebruiken in tal van (probleem)situaties en de aspecten die zinvol zijn in die specifieke situatie te kunnen identificeren en gebruiken. Leren detecteren welke vaardigheden uit CD nuttig zijn in een bepaalde situatie en welke niet, is net iets dat belangrijk is in CD.

Het idee om compartimentalisatie en fragmentatie te voorkomen klinkt misschien ook tegenstrijdig met decompositie als kernelement van CD. Dit is echter niet het geval. Onder “compartimentalisatie en fragmentatie voorkomen” verstaat men dat de opdrachten a priori niet te veel mogen opgedeeld zijn in (te) kleine delen [33]. Hierdoor zal de lerende immers nooit de hele puzzel leren leggen en zijn volledige taak kunnen uitvoeren. Decompositie als kernelement in CD is iets heel anders. De lerende wordt aangeleerd om een (complexe) taak te analyseren en op te delen in deelproblemen om daarna wél elk van deze deelproblemen op te lossen en tot een totaaloplossing te komen (zie 2.3.3).

4.2.2 Ontwerp van een les

4.2.2.1 Ontwerpstadia

Het ontwerpen van een les gebeurt in vier stappen. Eerst is er een goed lesidee nodig. Daarna volgt een mock-up, een soort schets van de les. In de derde stap wordt de les aan het leerplatform toegevoegd en volledig uitgewerkt. Tot slot is het belangrijk om de les grondig te testen en te laten testen door testgebruikers.

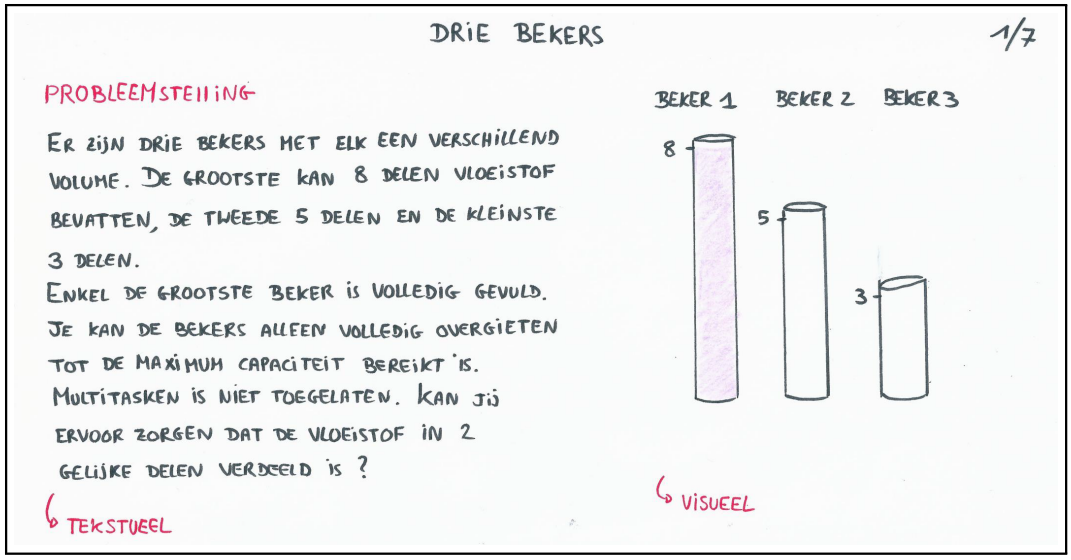
Idee
Een goed idee voor een les is zeer belangrijk. Vaak worden dan ook meerdere ideeën overwogen om er uiteindelijk slechts enkele goede over te houden. Een lesidee bestaat meestal uit een probleemstelling en één of meerdere oplossingsmethodes. Het is bij de keuze van het lesidee reeds belangrijk om rekening te houden met de elementen van CD die aan bod zouden kunnen komen in de les. Lesideeën waarbij de link met CD ver te zoeken is, zijn wellicht minder geschikt voor Co-De. Bij de uiteindelijke keuze van een idee voor een nieuwe les is het ook belangrijk om rekening te houden met de andere lessen op Co-De. Ideeën die focussen op nieuwe concepten en elementen van CD die nog niet zo vaak aan bod kwamen zijn interessant om verder te onderzoeken (zie ook tabel 4.1 op pagina 69).

In het kader van deze masterproef werd een werkdocument opgesteld met mogelijke lesideeën. Daaruit werden doorheen het academiejaar vijf ideeën geselecteerd om verder mee te werken. Deze lijst met ideeën is echter nog niet uitgeput en bevat nog heel wat onderwerpen die de moeite zijn om verder te onderzoeken (zie 5.3.2).

Mock-up
Na het selecteren van een idee wordt een mock-up gemaakt van de les. Deze mock-up is een soort van schets van de hele les. Het is de bedoeling om, op korte tijd, een goed zicht te krijgen op hoe de les er ongeveer gaat uitzien. De probleemstelling wordt uitgeschreven en de verschillende activiteiten worden bedacht. Ook de volgorde van de lesonderdelen, figuren en opdrachten worden geschetst. Het 4C/ID-model biedt een theoretisch kader om de volgorde te ontwerpen a.d.h.v. de moeilijkheidsgraad en variatie van de verschillende activiteiten (zie 4.2.1) [33].

Een mock-up kan worden gemaakt op papier of in een computerprogramma waarmee men vertrouwd is. Het loont de moeite om deze mock-up ook eens te laten bekijken door mensen die nog niet betrokken zijn in het proces. Zij kunnen aangeven of de les hen kan boeien, of de volgorde logisch is en of er zaken ontbreken. Vaak wordt de mock-up herhaaldelijk bijgestuurd en uitgebreid alvorens over te gaan naar de volgende stap.

De mock-ups voor de lessen op Co-De zijn voornamelijk gemaakt in Microsoft PowerPoint of op papier. Een stukje uit de handgeschreven mock-up van de les de drie bekers wordt afgebeeld in figuur 4.2 en de volledige mock-up van de les marslander, gemaakt in PowerPoint, is te vinden in appendix C.



Figuur 4.2: Een deel uit de handgeschreven mock-up voor de les de drie bekers

Implementatie
Nu volgt het uitwerken van de les op het leerplatform. Dit is de meest tijdsintensieve fase, maar men kan hier veel werk besparen door voldoende tijd te spenderen aan een goede mock-up. Binnen deze stap wordt een nieuwe les toegevoegd aan Co-De en worden alle secties en activiteiten toegevoegd aan die les. Ook afbeeldingen, animaties en programmeeropdrachten worden verder ontwikkeld en toegevoegd aan de les. Hierbij stuit men onvermijdelijk op nieuwe keuzes die nog moeten gemaakt worden en teksten die niet aan bod kwamen in de mock-up. Soms is het dan ook zinvol om even terug te keren naar de vorige stap en eerst de mock-up aan te passen en na te kijken.

Meer informatie over het bouwen van lessen, het toevoegen en bewerken van secties en activiteiten en de verschillende soorten activiteiten is terug te vinden in de gebruikershandleiding via `$code/handleiding`.

We bouwen nieuwe lessen meestal sectie per sectie op. Afgewerkte secties kunnen op die manier reeds worden getest vanuit het standpunt van de leerlingen. Onder deze fase valt ook het voorzien van informatie voor de leerkracht en het expliciet ingaan op de elementen van CD in de vorm van een feedback-activiteit zoals wordt beschreven in de uiteenzetting over de lesstructuur (zie 3.5.1.1).

Testen
Tot slot is het belangrijk om nieuwe lessen uitgebreid te testen en te laten testen. Het gedrag van de activiteiten verschilt op Co-De van rol tot rol. Het is dus cruciaal om afgewerkte lessen te testen vanuit het perspectief van de leerling. Het kan immers zo zijn dat deze komt vast te zitten in de loop van de les terwijl leerkrachten of beheerders zonder problemen verder kunnen. Deze testfase staat dan ook niet los van de vorige fases. Wanneer men bij het testen stuit op problemen of zaken die

toch beter anders aangepakt zouden worden, wordt de implementatie (en soms zelfs de mock-up) uit de vorige fases aangepast.

Ook inhoudelijk is het interessant om nieuwe lessen te laten testen. Uit zulke tests vloeien vaak zinvolle aanpassingen voort en ontstaat inspiratie voor uitbreidingen en variaties of voor nieuwe lessen. Regelmatig komen hier ook interessante ideeën voor alternatieve leerpaden uit voort.

4.2.2.2 Ontwerp van de leerpaden

Zoals reeds uitgebreid toegelicht in het vorige hoofdstuk (zie 3.5.1.1), hebben lessen op Co-De een vaste lesstructuur. De volgorde van de lesspecifieke inhoud wordt op het platform omschreven als een "leerpad". In de *(demo-)lessen* wordt telkens initieel uitgegaan van een standaard leerpad. De leerpaden werden echter zodanig geconcipieerd dat er gemakkelijk kan afgeweken worden van deze standaard, door delen anders te ordenen, inhoudelijk aan te passen, toe te voegen of weg te laten. Ook uit het 4C/ID-model volgt dat een zekere flexibiliteit in de leerpaden wenselijk is (zie 4.2.1.2).

Het standaard leerpad bestaat conceptueel uit twee delen: een probleemstelling en een zoektocht naar oplossingen. Deze delen hoeven elkaar niet sequentieel op te volgen. Meestal is de probleemstelling verweven met het zoeken en ontdekken van oplossingen voor concrete voorbeelden van het probleem. Vanuit deze concrete oplossingen wordt dan, over verschillende activiteiten heen, gezocht naar een algemene oplossingsmethode. De probleemstelling kan ook opgedeeld zijn in verschillende onderdelen waar eerst afzonderlijk wordt gezocht naar een oplossing. Dit is bijvoorbeeld het geval bij paardenronde & stadsgids, waar het probleem van de stadsgids niet samen met het probleem van de paardenronde wordt gepresenteerd.

De onderdelen uit een les zijn thematisch gegroepeerd in secties. Elke sectie vormt een afgewerkt deeltje, waardoor de les gemakkelijk herordend kan worden op het niveau van de secties. Het concept van secties op Co-De komt in grote mate overeen met een taakklas uit het 4C/ID-model dat beschreven wordt in 4.2.1. Zo kan het interessant zijn om, voor klasgroepen die meer ondersteuning nodig hebben, de sectie die een bepaald concept uitlegt voor de sectie met een opdracht daarover te plaatsen. Verder kunnen secties ook verborgen worden. In doolhoven bijvoorbeeld vormen de programmeeropdrachten voor zoekalgoritmes een sectie op zich. De leerkracht kan beslissen om die sectie weg te laten als er gewerkt wordt met een klas die nog geen programmeerervaring heeft.

In elke les wordt melding gemaakt van het standaard leerpad. Daarnaast staan er een aantal suggesties opgesomd van mogelijke keuzes betreffende een alternatief leerpad. De leerkracht wordt bewustgemaakt van de flexibiliteit binnen elke les. De alternatieve leerpaden zijn niet altijd volledig uitgewerkt. Indien een leerkracht beslist om voor een alternatief te kiezen, dan zal hij in zijn *eigen les* nog de secties en activiteiten moeten herordenen om het juiste leerpad te vormen (zie 4.2.4).

4.2.2.3 Ontwerp van de lesonderdelen

Bij het ontwerp van de individuele lesonderdelen wordt opnieuw een goede variatie in de soorten activiteiten beoogd. Om dit te illustreren, zijn er vier tabellen toegevoegd bij de bespreking van de lessen (tabel 4.2, 4.3, 4.4 en 4.5). Deze tabellen tonen de analyses van de uitgewerkte lessen. We omschrijven de verschillende eigenschappen van de activiteiten aan de hand van vier begrippen: *doe*, *denk*, *reflecteer* en *illustreer*.

Doe-activiteiten zijn lesonderdelen waarbij een bijzondere handeling wordt verricht (anders dan gewoon het doornemen van een stukje tekst of een vraag beantwoorden). Dit kan dus het uitwerken van een opdracht zijn, een animatie waarmee leerlingen moeten interageren, een oefening *unplugged* oplossen, iets op papier uittekenen...

Denk-activiteiten zijn lesonderdelen waar het uitdenken van een oplossing of antwoord belangrijk is, bijvoorbeeld wanneer naar een algoritme of oplossingsmethode moet worden gezocht.

Reflecteer-activiteiten koppelen bewust terug naar de handelingen uit vorige activiteiten. Zo moeten de leerlingen nadenken over vragen zoals “Hoe heb ik een bepaalde opdracht of vraag aangepakt?” of “Hoe moeilijk vond ik deze opdracht (in vergelijking met een andere opdracht)?”.

Illustreer-activiteiten ten slotte zijn activiteiten waarin een bepaald concept of een bepaalde methode wordt toegelicht. In paardenronde & stadsgids gebeurt dit bijvoorbeeld in het onderdeel “Zijn die opdrachten wel zo verschillend?” waar een animatie illustreert dat de paardenronde herleid kan worden tot de stadsgids.

Een evenwichtige les bevat elk van de vier besproken eigenschappen. Er volgt nu een korte bespreking over de samenstelling van elke les.

In paardenronde & stadsgids (tabel 4.2 op pagina 82) worden aan het begin van de les voornamelijk *doe*- en *denk*-activiteiten afgewisseld met *reflecteer*-activiteiten. Daarna zijn er meer *illustreer*-activiteiten verweven, omdat de concepten uit de *doe*- en *denk*-activiteiten dan verder kunnen worden toegelicht.

In de les doolhoven (tabel 4.3 op pagina 87) worden *doe*- en *denk*-activiteiten sneller afgewisseld met *illustreer*-activiteiten dan bij paardenronde & stadsgids. Dit is nodig omdat de heuristieken (“naïef zoeken” en “volg-een-muur”) en de zoekalgoritmes voor grafen (“diepte-eerst” en “breedte-eerst”) specifieke uitleg vereisen. Het tweede deel van doolhoven wordt gekenmerkt door twee grote programmeer-onderdelen. Die bevatten uiteraard *doe*- en *denk*-activiteiten. *Illustreer*-activiteiten zijn daar achterwege gelaten, aangezien er verwacht wordt dat leerlingen voordien al programmeerervaring hebben opgedaan. Mocht de les gegeven worden zonder de programmeeropdrachten in secties 7 & 8 (zie 4.4.4), dan zou het wellicht nodig zijn om bijkomende *doe*- en *denk*-activiteiten te voorzien, want die zijn er niet in secties 4-6. Leerlingen zouden dan niet voldoende kunnen nagaan of ze de zoekalgoritmes correct kunnen toepassen.

De lesinhoudelijke secties uit de marslander (tabel 4.4 op pagina 93), met uitzondering van de inleiding, bevatten telkens de vier soorten activiteiten. Elke sectie gaat in op een specifiek algoritme. Aan elk algoritme zijn één of meerdere opdrachten verbonden (*doe* en *denk*). De werking van het algoritme wordt ook uitgelegd (*illustreer*). Verder moet ook nagedacht worden over de eigenschappen van de verschillende algoritmes. De leerlingen moeten daarvoor hun denkproces bij de opdrachten in beschouwing nemen (*reflecteer*). De programmeeropdrachten zitten, in tegenstelling tot bij doolhoven, verspreid over de drie secties.

In de drie bekers (tabel 4.5 op pagina 97) zijn er twee lesinhoudelijke onderdelen, namelijk sectie 2 "opdracht" en sectie 3 "graafvoorstelling". Sectie 2 focust op het vinden van een oplossing voor de puzzel van de drie bekers zonder verdere uitleg. Hierin zitten dus voornamelijk *doe*- en *denk*-opdrachten. Het denkproces rond die opdrachten wordt bevraagd door middel van enkele *reflecteer*-opdrachten zoals bij de andere lessen op het platform. Sectie 3 behandelt een meer gestructureerde manier om de puzzel op te lossen met behulp van een graafvoorstelling. Die voorstelling wordt stap voor stap uitgewerkt aan de hand van een aantal *illustreer*-activiteiten. De werkwijze uit sectie 3 leren de leerlingen toepassen op de variaties in sectie 5 "variaties & uitbreidingen".

De vier lessen bevatten allemaal de vier leseigenschappen. Wel is de interne organisatie van die eigenschappen niet dezelfde in elke les. De manier van aanpak is afhankelijk van de specifieke probleemstelling in een les. Wanneer nieuwe lessen worden gecreëerd, zijn de vier eigenschappen een goede leidraad om de les vorm te geven.

4.2.3 Soorten lessen

Zoals reeds werd beschreven in het vorige hoofdstuk bestaan er op Co-De twee soorten lessen, namelijk *demo-lessen* en *eigen lessen*.

Demo-lessen zijn lessen die iedereen kan volgen en die beheerd worden door de beheerders van Co-De. Ze kunnen gebruikt worden zonder enige configuratie maar kunnen niet worden aangepast door leerkrachten. *Eigen lessen* zijn lessen voor een bepaalde leerkracht of school. De desbetreffende leerkracht kan deze lessen aanpassen zoals verderop wordt beschreven (zie 4.2.4).

Meer uitleg over het verschil tussen beide soorten is terug te vinden in onderdeel 3.5.1.3 en in tabel 3.2 (p. 57). Voor gebruikers van Co-De wordt het verschil uitgebreid toegelicht in de handleiding[1].

[1] `$code/mod/page/view.php?id=206`.

4.2.4 Aanpasbaarheid van een les

4.2.4.1 Demo-les

Een *demo-les* toont uitsluitend het voorziene standaard leerpad. Voor een demo-leerkracht is het niet mogelijk hieraan aanpassingen te doen. De demo-leerkrachten kunnen wel de suggesties zien voor alternatieve leerpaden bij "informatie voor leerkrachten".

4.2.4.2 Eigen les

In *eigen lessen* zijn tal van manieren om de les aan te passen voorzien. Uiteraard is dit enkel toegelaten voor de leerkracht van de betreffende les. Er volgt een korte opsomming van de verschillende mogelijkheden. Een meer gedetailleerde beschrijving en stappenplan vindt men in de gebruikershandleiding[2].

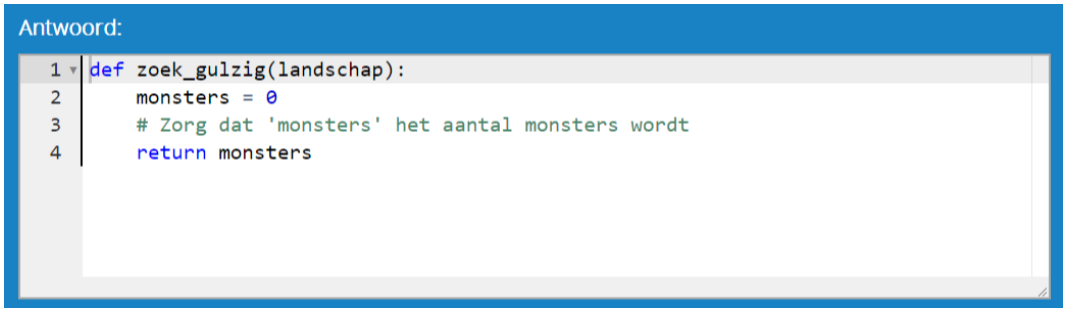
Leerpad configureren
Op het hoogste niveau kan het leerpad aangepast worden door secties uit de les te verbergen, te tonen, toe te voegen, te verwijderen of van plaats te wisselen. Daarnaast is het ook mogelijk om individuele activiteiten te verbergen, te tonen, toe te voegen, te verwijderen of van plaats te wisselen.

Niveau van ondersteuning
Bij een aantal activiteiten en activiteiten-soorten is het mogelijk om het niveau van ondersteuning in te stellen. Zo bestaat de animatie van de paardenronde (zie tabel 4.2 op pagina 82) in twee gedaantes: de standaard opdracht en een gewijzigde versie die meer visuele ondersteuning biedt door telkens aan te geven waar het paard al gepasseerd is. De leerkracht heeft de keuze om één van de twee of beide versies te tonen. Ook in de testen kan meer ondersteuning geboden worden indien nodig. Zo is het mogelijk om tekstuele hints toe te voegen. Voor de programmeeropdrachten heeft de cursusbouwer of leerkracht de vrijheid om te kiezen welk vooraf geschreven skelet van broncode wordt aangeboden aan de leerlingen (zie afbeelding 4.3).

Deze keuzes komen ook aan bod bij het 4C/ID-model bij het ontwerpen van de ondersteunende en procedurele informatie (zie 4.2.1.1). Een goed ontwerp hiervan hangt af van de specifieke context van de klas en de leerkracht. Daarom worden op Co-De verschillende versies en mogelijkheden aangeboden om te variëren in deze ondersteuning.

Activiteiten en secties aanpassen
Het is ook mogelijk om de inhoud en instellingen van bestaande secties en activiteiten aan te passen. Samen met het instellen van het leerpad geeft deze mogelijkheid volledige controle over de les aan de desbetreffende leerkracht.

[2] `$code/handleiding`.



Figuur 4.3: Het antwoordskelet bij een programmeeropdracht uit de marslander. De cursusbouwer of leerkracht kan instellen welke programmacode de leerling al dan niet te zien krijgt bij het oplossen van de opgave. Zowel de hoofding van de methode als de variabele die nog een juiste toekenning moet krijgen zijn hier al gegeven.

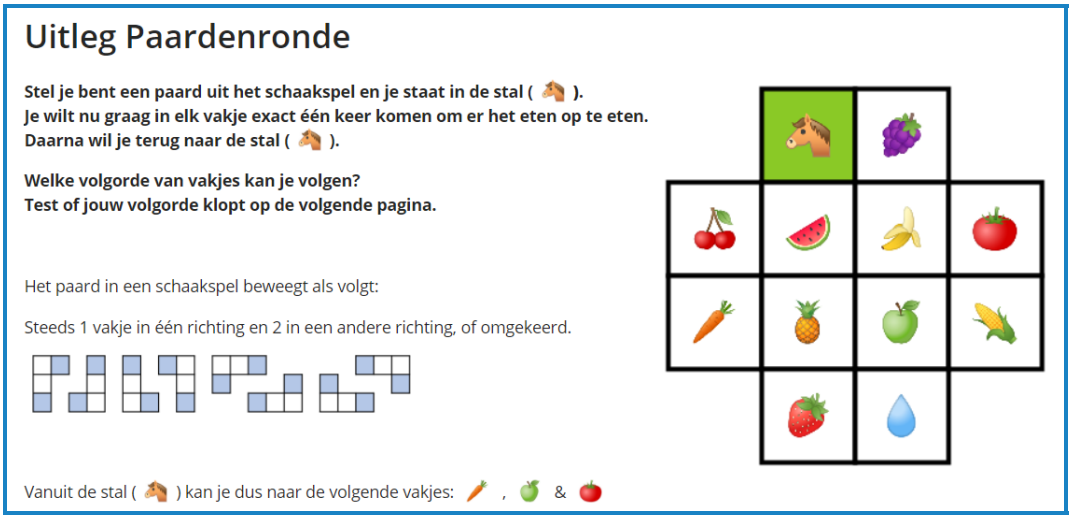
4.3 Les 1: de paardenronde en de stadsgids

De demo van de deze les is beschikbaar via `$code/course/view.php?id=3`.

Deze les is gebaseerd op een idee dat Paul Curzon beschrijft doorheen hoofdstuk vijf van zijn boek "The Power of Computational Thinking" [12, p. 65-80]. De les werd ook uitgebreid getest. De resultaten van deze test zijn terug te vinden in appendix B en worden besproken in 3.7.1 (p. 66).

4.3.1 Probleemstelling

Deze les presenteert de paardenronde, een moeilijk probleem, dat meestal via trial-and-error wordt opgelost. Bij de paardenronde moet men een route proberen te vinden voor het paard uit het schaakspel. In deze route moet het paard overal exact éénmaal komen en dan terug op zijn beginpositie belanden (zie figuur 4.4).



Figuur 4.4: De eerste opgave in de les paardenronde & stadsgids.

Daarna wordt een ander, eenvoudiger en meer intuïtief probleem gegeven in de vorm van een stadsgids. Nu moet men een route uitstippelen op een metrokaart voor een stadsgids die een groep toeristen rondleidt. Zij starten aan het hotel, willen overal exact éénmaal passeren, stappen steeds van de metro af na één halte en willen terug eindigen aan het hotel (zie figuur 4.5).

Figuur 4.5: De tweede opgave in de les paardenronde & stadsgids.

4.3.2 Standaard leerpad

In het standaard leerpad (zie tabel 4.2) krijgen de leerlingen eerst het probleem van de paardenronde en daarna het probleem van de stadsgids voorgeschoteld. Ze gaan op zoek naar een oplossing aan de hand van een animatie die hen feedback geeft over de correctheid van hun oplossing (zie 3.5.2.2). Daarna worden ze bevraagd over hoe moeilijk ze het vonden en hoe ze (al dan niet) tot een oplossing gekomen zijn.

Na het evalueren van beide oefeningen start een klassikaal gesprek of een discussie in groepjes over de moeilijkheidsgraad van beide opdrachten en de methodes waarmee leerlingen tot een oplossing kwamen. Er wordt ook gevraagd om op zoek te gaan naar verschillen en gelijkenissen tussen de twee opgaven.

Daarna wordt duidelijk gemaakt dat de twee oefeningen eigenlijk zeer gelijkend zijn en een oplossing van het ene probleem kan omgezet worden in een oplossing van het andere probleem. Het eerste moeilijke probleem kan eigenlijk worden opgelost door het als een graaf voor te stellen, zoals dat reeds het geval is bij de tweede opgave. Daarna zijn beide problemen op te lossen door een Hamiltoniaanse kring te zoeken in een ongerichte graaf[3]. De overgang van het ene probleem naar het andere wordt voorgesteld aan de hand van een interactieve animatie. Daarna volgt een kleine uitbreiding van de oefening, die voor de paardenronde terug veel denkwerk vraagt, maar bij de stadsgids op het zicht kan worden opgelost.

Tot slot wordt uitgelegd wat er juist gebeurde, welke concepten en vaardigheden aan bod kwamen en waarom dit zinvol is. Ook enkele open vragen geven stof tot verder nadenken. In deze vorm neemt de les ongeveer **twee lesuren** in beslag.

[3] Meer uitleg over Hamiltoniaanse kringen via https://www.tijdschriftkarakter.be/het-probleem-van-de-handelsreiziger/.

PAARDENRONDE & STADSGIDS		Doe	Denk	Reflecteer	Illustreer
1 DE PAARDENRONDE	sectie				
Uitleg Paardenronde	pagina	+	+		
Opdracht: zoek een volgorde van vakjes*	animatie	+	+		
Hoe moeilijk vond je de paardenronde?	keuze			+	
Welke technieken heb je gebruikt?	keuze			+	
Evaluatie: de paardenronde	test		+	+	
2 DE STADSGIDS	sectie				
Uitleg Stadsgids	pagina	+	+		
Opdracht: zoek een volgorde van bezienswaardigheden	animatie	+	+		
Hoe moeilijk vond je het stadsgidsprobleem?	keuze			+	
Welke technieken heb je gebruikt?	keuze			+	
Evaluatie stadsgids	test		+	+	
3 GELIJKENISSEN EN VERSCHILLEN	sectie				
Welke opdracht vond jij de moeilijkste?	keuze			+	
Gelijkenissen en verschillen	test		+	+	
4 VERSCHILLENDE WEERGAVEN	sectie				
Zijn die opdrachten wel zo verschillend?	animatie		+		+
Twee weergaven	pagina	+	+		
Overeenkomende vakjes & bezienswaardigheden	pagina				+
5 TWEE VAKJES EXTRA	sectie				
Opdracht: twee vakjes extra	pagina	+	+		
Uitleg oplossen twee vakjes extra	pagina				+
6 COMPUTATIONEEL DENKEN	sectie				
Elementen computationeel denken	feedback			+	
7 VARIATIES & UITBREIDINGEN	sectie				
Kleine variaties	pagina	+	+		
Kleine variaties: meer info	pagina				+
Bestaat er altijd een volgorde?	pagina	+	+		
Bestaat er altijd een volgorde? Meer info	pagina				+
Meerdere volgordes	pagina	+	+		
Meerdere volgordes: meer info	pagina				+
8 EINDE	sectie				
Afsluiter	pagina		+		+

* Verschillende versies zijn beschikbaar met verschillende niveaus van ondersteuning.

Tabel 4.2: Het standaard leerpad van de les paardenronde & stadsgids met de eigenschappen van de verschillende activiteiten en snelkoppelingen naar de verschillende secties.

4.3.3 Elementen computationeel denken

In deze les komen voornamelijk de elementen *abstractie* en *veralgemening* van computationeel denken aan bod. De volledige bespreking van de kernelementen van CD in deze les is online terug te vinden[4].

Abstractie

Abstractie is voornamelijk terug te vinden in het gedeelte van de les waarin blijkt dat men een oplossing voor het ene probleem kan transformeren in een oplossing voor het andere. Hierbij wordt in de paardenronde abstractie gemaakt van het probleem door de symbolen achterwege te laten en ze te vervangen door knopen, gelabeld met letters. Daarnaast worden de ingewikkelde stapregels van het paard omgezet in bogen tussen de knopen. Nu is de paardenronde voorgesteld als een soortgelijk probleem als de stadsgids.

Ook bij het oplossen van de stadsgids wordt *abstractie* gebruikt. De symbolen worden ook daar achterwege gelaten en enkel de relevante informatie (knopen en bogen) wordt gebruikt voor het vinden van een route. Gegevens over hoe de monumenten eruitzien, hoe oud ze zijn, met hoeveel de groep is en dergelijke meer zijn hier niet relevant.

Tot slot komt *abstractie* ook aan bod in het oplossen van de opgave met twee vakjes extra (zie tabel 4.2). Ook hier is het belangrijk om enkel de relevante informatie te gebruiken. Daarenboven loont het de moeite om te vertrekken van de graaf uit de vorige oefeningen en te focussen op de twee nieuwe vakjes in plaats van het hele denkproces opnieuw te moeten doorlopen.

Veralgemening

Ten eerste wordt door een opgave te geven met twee vakjes extra een nieuwe instantie van hetzelfde probleem voorzien. Op die manier wordt de oplossingsmethode meerdere malen gebruikt en leert de leerling deze zelf toe te passen. Het leren gebruiken van de oplossingsmethode op nieuwe problemen van dezelfde soort maakt integraal deel uit van *veralgemening*.

Ten tweede is ook het omzetten van een probleem in een graaf om daarin een oplossing te zoeken, een algemene aanpak. Er wordt in deze les niet expliciet ingegaan op andere problemen waar grafen zinvol kunnen zijn. Door dit proces echter herhaaldelijk tegen te komen in verschillende lessen wordt dit ook bevattelijk.

4.3.4 Ontwerpkeuzes

Volgorde van de twee probleemstellingen

Een eerste belangrijke ontwerpkeuze is om binnen het standaard leerpad eerst de paardenronde te behandelen en daarna de stadsgids. Deze schikking is bewust zo gekozen omdat dan het effect het grootst is wanneer men komt bij de uitleg dat de twee eigenlijk zeer gelijkend zijn. Door de moeilijkste opdracht als eerste aan te bieden worden de leerlingen ook meteen uitgedaagd en geboeid. Het is echter

[4] `$code/mod/feedback/view.php?id=21`.

ook mogelijk om de volgorde van deze twee secties om te draaien en zo eerst een succeservaring te creëren door de gemakkelijkste opgave eerst te plaatsen.

Uitgebreide reflectie vooraan in de les
Bij beide opgaven wordt er veel aandacht besteed aan reflectie. Zowel de moeilijkheidsgraad, de gebruikte hulpmiddelen als de algemene oplossingsmethode worden bij beide bevraagd. Het is hierbij de bedoeling om een gesprek tussen de leerlingen te starten en ze reeds zelf te laten ontdekken dat er een zekere gelijkenis bestaat tussen de twee problemen. Daarom werd ook gekozen om aan de leerlingen de antwoorden van de klas te tonen in taartdiagrammen. Op die manier kunnen de leerlingen hun antwoord kaderen binnen de antwoorden van de klasgroep. Dit biedt een aanleiding om een gesprek over de antwoorden te starten. De verschillende *reflecteer*-activiteiten kunnen worden verborgen door de leerkracht indien gewenst.

Interactieve animaties bij de opgaven
Om vertrouwd te geraken met de problemen en om op zoek te gaan naar een oplossing voor een specifieke opgave werden enkele interactieve animaties toegevoegd aan de les. Na de uitleg van beide opgaven krijgt de leerling zo een animatie voorgeschoteld waarin hij of zij kan proberen een oplossing te vinden. De animatie geeft feedback over de aangeboden oplossing en laat een onbeperkt aantal pogingen toe. Voor de paardenronde werden verschillende versies gebouwd met verschillende niveaus van ondersteuning. Via deze interactieve animaties kan men doorgaans sneller progressie maken dan volledig op papier te moeten werken en is ook specifieke feedback mogelijk.

Interactieve animatie die de overgang duidelijk maakt
Ook bij het uitleggen dat de twee problemen met elkaar verwant zijn, wordt een animatie gebruikt. Op deze manier kan elke leerling de stappen van de omzetting op zijn of haar eigen tempo doorlopen en herhalen. Een tweede grote meerwaarde is dat bewegende elementen mogelijk zijn. Elementen die worden verplaatst, bewegen echt op het scherm en kan men gedurende de hele tijd duidelijk volgen. Dit is niet mogelijk in een stappenplan via afbeeldingen, door een beschrijving of in een boek.

Toevoeging van twee vakjes extra
In het standaard leerpad volgt, na het samenbrengen van de twee opgaven, een uitbreiding van het probleem. De bedoeling hiervan is om zelf actief aan de slag te gaan met het verworven inzicht. Bij deze uitbreiding is er geen animatie voorzien, maar wordt er gewerkt op papier, vertrekkende van de graaf uit de initiële opgave.

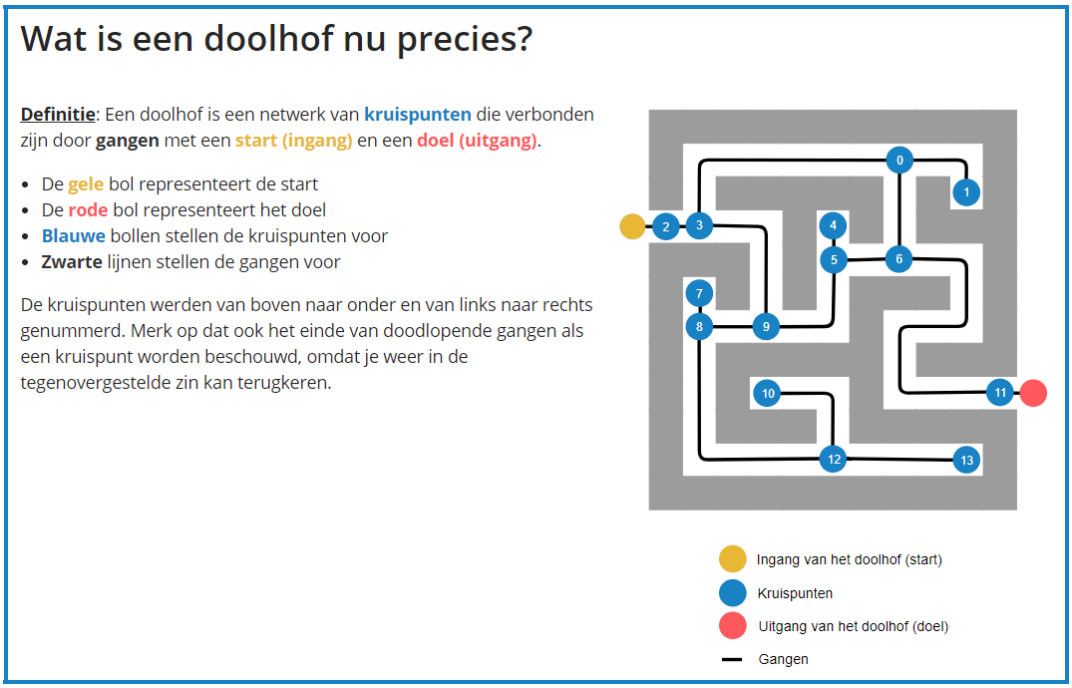
Alternatieve leerpaden bieden twee interessante opties in verband met de toevoeging van de twee vakjes extra. Ten eerste kan men als leerkracht beslissen om eerst het probleem uit te breiden, nog voor de transformatie uit de doeken wordt gedaan. Sommige klasgroepen zullen immers zelf reeds tot dit inzicht gekomen zijn bij het oplossen van de twee problemen en het zoeken naar gelijkenissen en verschillen. Een tweede gangbare optie is om deze uitbreiding over te slaan. Dit kan bijvoorbeeld uit tijdsoverwegingen of om klassikaal in te gaan op een andere uitbreiding of variatie.

4.4 Les 2: doolhoven

De demo van de deze les is beschikbaar via `$code/course/view.php?id=5`.

4.4.1 Probleemstelling

De les doolhoven handelt over het concept doolhoven ("Wat is een doolhof?") en verschillende strategieën, heuristieken en algoritmes om op een plattegrond van een doolhof een pad te vinden van de in- naar de uitgang (zie afbeelding 4.6).



Figuur 4.6: De definitie van een doolhof uit de les doolhoven.

In de optionele uitbreiding van de les wordt er teruggegrepen naar het probleem van de paardenronde (zie 4.3). Zoekalgoritmes voor doolhoven kunnen ook daarop worden toegepast.

4.4.2 Standaard leerpad

De les start met een inleiding waarin de probleemstelling wordt uitgelegd (sectie 1 op tabel 4.3). Leerlingen worden aangemoedigd om methodes te bedenken die een pad zullen vinden in allerlei doolhoven. Er is een downloadbaar werkblad[5] voorzien waarop ze ideeën kunnen uittesten.

Secties 2 tot en met 6 gaan dieper in op enkele oplossingsmethodes. Daarvoor wordt eerst uitgelegd wat een algoritme is. De les doolhoven is namelijk de eerste waarin dit begrip aan bod komt. Dezelfde sectie bespreekt ook wat een algoritme voor doolhoven is (in dit geval): men wil een pad kunnen vinden van de ingang naar elke andere positie in het doolhof, in eindige tijd.

Daarna volgt een eerste methode om een pad te vinden in een doolhof (sectie 3), genaamd "volg-een-muur". Volg-een-muur is een zeer gemakkelijke methode maar ze zal niet altijd een pad naar de uitgang vinden. Aan de hand van een aantal voorbeelden zullen de leerlingen ontdekken dat deze methode enkel een pad vindt als alle muren van de doolhof met elkaar in verbinding staan. Er is dus nood aan andere methodes om een pad te vinden.

Die andere methodes kunnen enkel aangebracht worden indien er met een uniforme voorstelling van de doolhoven wordt gewerkt. Daarom gaat sectie 4 dieper in op het concept "grafen". Er wordt getoond hoe elk doolhof getransformeerd kan worden in een graaf (afbeelding 4.7).

In sectie 5 komen daarna twee zoekalgoritmes voor grafen aan bod: "diepte-eerst" en "breedte-eerst". Beide algoritmes vinden steeds een pad naar de uitgang maar gedragen zich zeer verschillend. Sectie 6 maakt die verschillen duidelijk aan de hand van enkele gerichte vragen.

Sectie 7 en 8 bevatten uitsluitend programmeeropdrachten die de graafvoorstelling, het diepte-eerst algoritme en het breedte-eerst algoritme omsluiten. Die opdrachten zijn opgesplitst in deelopdrachten. Ze gaan dus steeds verder met het resultaat van de vorige opdracht maar zijn zo opgesteld dat het ook mogelijk is om een deelopdracht op zichzelf op te lossen.

Verder is er de standaard vragenlijst over computationeel denken die vervat zit in sectie 9. Vervolgens zijn er nog een aantal uitbreidingen in verband met de aangeleerde zoekalgoritmes. Die algoritmes zijn bijvoorbeeld ook toepasbaar op problemen uit andere lessen, zoals paardenronde & stadsgids. Op het einde van de les is er nog een korte afsluiter.

In deze vorm neemt de les ongeveer **twee tot vier lesuren** in beslag.

[5] Via `$code/resources/doolhoven/werkblad_doolhoven1.pdf`.

DOOLHOVEN		Doe	Denk	Reflecteer	Illustreer
1 INLEIDING	sectie				
Uitleg	pagina	+	+		
2 ALGORITMEN	sectie				
Wat is een algoritme?	pagina				+
Algoritmes voor doolhoven	pagina		+		+
3 VOLG-EEN-MUUR	sectie				
Opdracht: volg-een-muur	pagina	+	+		+
Denk je dat volg-een-muur altijd werkt?	keuze		+	+	
Enkelvoudig samenhangende ruimtes	pagina				+
Andere algoritmes voor doolhoven	test	+	+		
4 WAT IS EEN DOOLHOF?	sectie				
Wat is een doolhof nu precies?	pagina		+		+
Wat is een doolhof nu precies? (2)	pagina				+
Graafvoorstelling	pagina				+
5 ZOEKALGORITMES VOOR GRAFEN	sectie				
Inleiding: zoekalgoritmes voor grafen	pagina		+		+
Het diepte-eerst zoekalgoritme	pagina				+
Het breedte-eerst zoekalgoritme	pagina				+
6 ZOEKALGORITMES EVALUEREN	sectie				
Zoekalgoritmes evalueren	test		+	+	
Feedback: zoekalgoritmes evalueren	pagina		+		+
7 PROGRAMMEREN: GRAAFVOORSTELLING	sectie				
Programmeeropdracht: knopen	test	+	+		
Programmeeropdracht: bogen	test	+	+		
Programmeeropdracht: graafvoorstelling	test	+	+		
8 PROGRAMMEREN: ZOEKALGORITMES	sectie				
Programmeeropdracht: diepte-eerst	test	+	+		
Programmeeropdracht: breedte-eerst	test	+	+		
9 COMPUTATIONEEL DENKEN	sectie				
Elementen computationeel denken	feedback			+	
10 VARIATIES & UITBREIDINGEN	sectie				
Variaties & uitbreidingen	pagina	+	+		+
11 EINDE	sectie				
Afsluiter	pagina				+

Tabel 4.3: Het standaard leerpad van de les doolhoven met de eigenschappen van de verschillende activiteiten en snelkoppelingen naar de verschillende secties.

4.4.3 Elementen computationeel denken

In deze les komen voornamelijk de elementen *abstractie*, *algoritmisch denken* en *evaluatie* van computationeel denken aan bod. De volledige bespreking van de kernelementen van CD in deze les is online terug te vinden[6].

Abstractie

Er wordt *abstractie* gemaakt van de lay-out van de doolhof. De leerlingen proberen uit te zoeken welke gegevens echt noodzakelijk zijn om op een gestructureerde manier de uitgang van de doolhof te vinden. Die elementen kunnen herleid worden tot een graafvoorstelling (afbeelding 4.7).

Algoritmisch denken

Bij het volg-een-muur-principe en de zoekalgoritmes voor grafen is *algoritmisch denken* van groot belang. De stappen in de zoekalgoritmes liggen eenduidig vast en moeten in de juiste volgorde gevolgd worden. De leerlingen leren om een dergelijk stappenplan te interpreteren en te analyseren. Daarnaast is doolhoven de les bij uitstek om grafen te programmeren. Ook diepte-eerst en breedte-eerst zijn vervat in twee programmeeropdrachten. Deze algoritmes vormen een goede basis om zich nadien te verdiepen in andere algoritmes om bomen of grafen te doorzoeken, bijvoorbeeld om het kortste pad te vinden.

Evaluatie

Dit kernelement van CD komt op verschillende plaatsen in de les aan bod. Ten eerste wordt de werking van volg-een-muur geanalyseerd. Die methode vindt enkel oplossing bij enkelvoudig samenhangende ruimtes. Ten tweede worden de zoekalgoritmes diepte-eerst en breedte-eerst onderling met elkaar vergeleken.

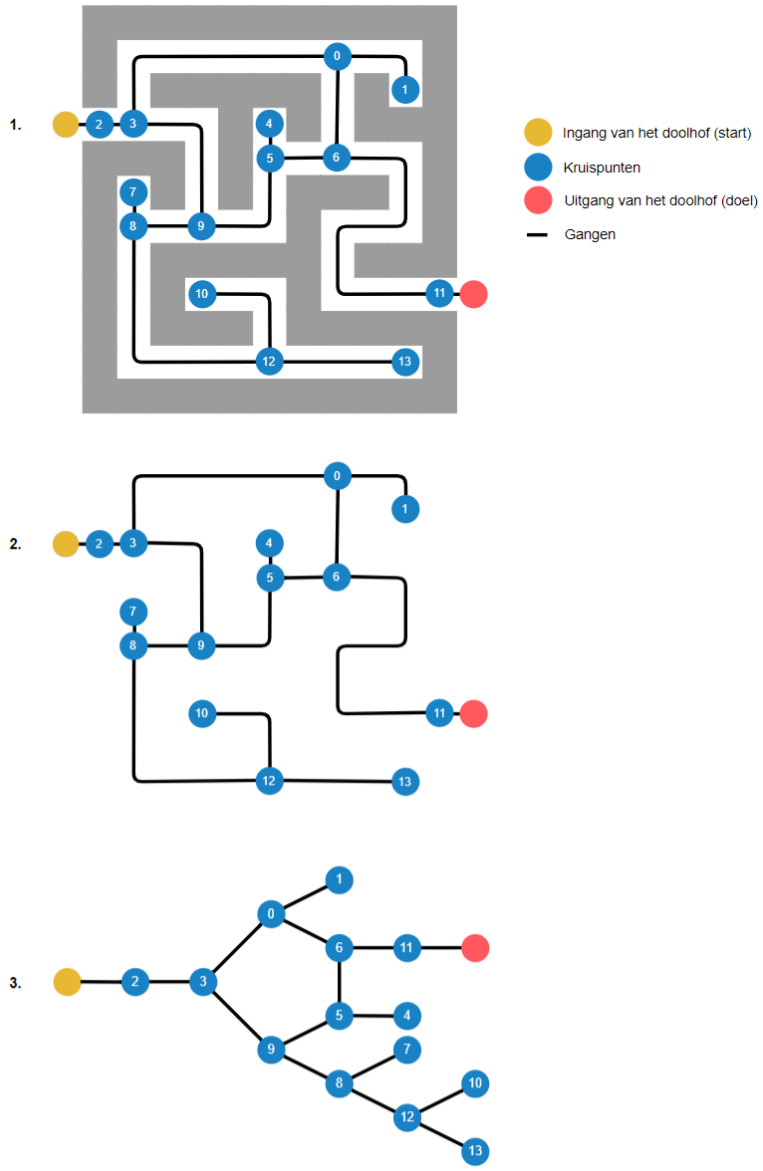
[6] `$code/mod/feedback/view.php?id=54`.

Graafvoorstelling

Om een oplossing voor ons vraagstuk te vinden, is het voldoende om enkel de elementen uit onze definitie over te houden. Wanneer we dat doen, spreken we over een **graaf** die bestaat uit **knopen** en **bogen.**

Aangezien we de bogen (of gangen) in twee richtingen kunnen bewandelen, wordt ons doolhof een **ongerichte graaf** (en geen gerichte graaf). Het maakt niet meer uit hoe we de graaf tekenen, zolang de relaties tussen de knopen bewaard blijft. Daarom is het vaak overzichtelijker om de grafen als een hiërarchische structuur te tekenen:

1. **Benoem de elementen uit de definitie**
2. **Laat de overbodige gegevens weg**
3. **Optioneel: herschik de bogen en knopen**



Figuur 4.7: Het herleiden van een doolhof tot zijn graafvoorstelling uit de les doolhoven.

4.4.4 Ontwerpkeuzes

Programmeeropdrachten
De programmeeropdrachten zijn in deze les verdeeld in twee secties. Zo wordt de graafvoorstelling apart van de zoekalgoritmes behandeld. De leerkracht kan zo gemakkelijk instellen of hij geen, één of beide secties wil behandelen. Binnen elke sectie zijn de programmeeropdrachten opnieuw opgesplitst in verschillende deeltaken. Zo kan opnieuw gemakkelijk de volgorde worden aangepast en kunnen één of meerdere opdrachten verborgen worden. De opgavebestanden staan telkens bij de opdrachten zodat er ook met een eigen lokale programmeeromgeving gewerkt kan worden.

Volledige uitwerking van de zoekalgoritmes
In deze les wordt van elk zoekalgoritme een volledig stappenplan getoond. Een uitgewerkt voorbeeld geeft alle iteraties van elk algoritme weer, totdat de doelknoop bereikt is. Bij elke iteratie worden ook de stappen uit het stappenplan getoond die van toepassing zijn. Die volledige uitwerking maakt het mogelijk voor leerlingen om de algoritmes zelfstandig te verwerken op hun eigen tempo.

Andere zoekalgoritmes
In doolhoven wordt enkel ingegaan op diepte-eerst en breedte-eerst. Dit zijn twee bekende algoritmes die altijd zullen werken (indien er een pad tussen start- en doelknoop bestaat), maar niet noodzakelijk een optimaal pad vinden. De les geldt dus als een goede introductie rond zoekalgoritmes voor grafen, maar zou nadien kunnen gevolgd worden door een les over andere algoritmes die altijd het kortste pad vinden. Dit werd bewust niet meer geïntegreerd in deze les omdat de les nu al meer dan één lesuur in beslag neemt.

Verwijzing naar paardenronde & stadsgids
In de uitbreidingen & variaties van doolhoven wordt er verwezen naar het probleem van paardenronde & stadsgids. Dankzij de graafvoorstelling voor de paardenronde kunnen de zoekalgoritmes (diepte-eerst, breedte-eerst) uit doolhoven ook toegepast worden op de graaf van die les. Zo wordt meer coherentie gecreëerd tussen de verschillende lessen. De nieuwe toepassing vraagt natuurlijk wel enige aanpassing van het algoritme en de implementatie bij de programmeeropdrachten. De uitbreiding is daarom optioneel. Er werd geen modeloplossing uitgewerkt en er is geen kant-en-klare programmeeropdracht beschikbaar voor deze uitbreiding.

4.5 Les 3: de marslander

De demo van de deze les is beschikbaar via `$code/course/view.php?id=9`.

4.5.1 Probleemstelling

In deze les over de marslander gaan we op zoek naar een reisweg voor een rover op Mars. De bedoeling is om de beste reisweg te vinden, namelijk diegene waarop de rover de meeste bodemstalen (monsters) kan verzamelen. De rover bevindt zich op een vierkant raster. Hij start steeds in de noordwestelijke hoek, moet eindigen in de zuidoostelijke hoek en kan zich enkel bewegen naar het oosten en het zuiden. Figuur 4.8 toont de probleemstelling zoals deze gepresenteerd wordt aan de leerlingen na een korte introductie over de marsmissies van NASA.

Het doel van deze les is om algoritmes te vinden en te implementeren die de beste reisweg kunnen opsporen. Hierbij is er heel wat aandacht voor de voor- en nadelen van de verschillende algoritmes en kan de leerkracht kiezen om het programmeren achterwege te laten.

Figuur 4.8: De opgave uit de inleiding van de les over de marslander.

4.5.2 Standaard leerpad

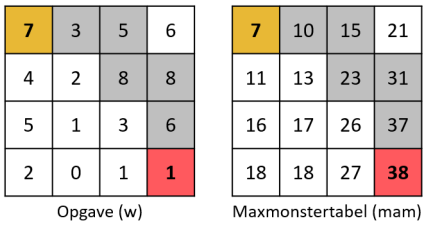
Het standaard leerpad (zie tabel 4.4) start met een inleiding die bestaat uit vier delen. Eerst komt een introductie met achtergrondinformatie en een filmpje van de eerste marsmissie van NASA. Daarna wordt de eigenlijke probleemstelling gepresenteerd (zie 4.5.1). Als derde volgen verschillende interactieve animaties waarin telkens voor een bepaald marslandschap moet worden gezocht naar de beste route. Deze marslandschappen verschillen van grootte en komen van klein naar groot aan bod. In die animatie krijgen de leerlingen feedback over de correctheid, volledigheid en optimaliteit van hun oplossing. Tot slot wordt kort toegelicht wat er in de rest van de les aan bod zal komen en waar de leerlingen moeten op letten.

Eerst wordt een exhaustief zoekalgoritme onderzocht. Hierbij worden alle mogelijke reiswegen bestudeerd om uiteindelijk de beste reisweg te vinden. In deze sectie is aandacht voor hoeveel reiswegen er bestaan voor een marslandschap van vaste grootte (vier op vier) en voor marslandschappen van willekeurige groottes. Hierbij komt ook een formule uit kansrekenen aan bod. Er wordt besloten dat dit zoekalgoritme zeker de optimale reisweg zal vinden, maar daar (voor grotere marslandschappen) behoorlijk lang zal over doen.

Een tweede algoritme dat wordt aangehaald, is een gulzig zoekalgoritme. Dit algoritme construeert maar één reisweg en is dus veel sneller dan het vorige. Het doet dit door steeds (lokaal) de beste optie te kiezen tussen oost en zuid. Eens het principe duidelijk is, wordt gevraagd om het gulzig zoekalgoritme toe te passen op enkele voorbeelden en na te denken over de optimaliteit van de gevonden oplossing. Zo komt men tot het inzicht dat deze methode niet altijd de best mogelijke reisweg vindt. Dit komt omdat men kan komen vast te zitten in lokale maxima, ook dat wordt toegelicht. Tot slot is er ruimte om dit algoritme te implementeren. Dit is een leuke en vrij eenvoudige programmeeropdracht.

Daarna wordt stap-voor-stap een derde algoritme opgesteld dat probeert om de tekortkomingen van beide voorgaande algoritmes weg te werken. Dit dynamisch-geprogrammeerd algoritme[7] stelt een *maxmonstertabel*[8] (*mam*) op via een recursieve vergelijking. Een voorbeeld van een marslandschap en de corresponderende *mam* wordt afgebeeld op figuur 4.9. Op de *mam* kan men het maximum te behalen aantal monsters aflezen in de zuidoostelijke hoek. De beste reisweg kan men ten slotte terugvinden door het gulzig zoekalgoritme in omgekeerde volgorde toe te passen op de *mam*, vertrekkend van de zuidoostelijke hoek. Opnieuw wordt aan de leerlingen gevraagd om dit algoritme zelf toe te passen op een aantal voorbeelden en hun bevindingen te beschrijven. Daarna volgen twee programmeeropdrachten. De eerste dient om de *mam* op te stellen en de tweede om de beste reisweg te vinden.

Na de lesspecifieke inhoud volgt de feedback-activiteit over de elementen van computationeel denken en worden er nog enkele open uitbreidingen en variaties aangeboden. In deze vorm neemt de les ongeveer **twee à drie lesuren** in beslag.



Figuur 4.9: Een voorbeeld van marslandschap (links) en de corresponderende maxmonstertabel (rechts) uit de marslander.

[7] Meer uitleg over dynamisch programmeren: `$code/mod/page/view.php?id=157` of `https://en.wikipedia.org/wiki/Dynamic_programming`.
[8] `$code/mod/page/view.php?id=166`.

DE MARSLANDER		Doe	Denk	Reflecteer	Illustreer
1 INLEIDING	sectie				
NASA's Sojourner	pagina				+
Filmpje: leef je in	pagina				+
Een reisweg voor de rover	pagina	+	+		
Opdracht: zoek de beste reisweg (4x4m)	pagina	+	+		
Opdracht: zoek de beste reisweg (6x6m)	pagina	+	+		
Algoritmes voor de marslander	pagina		+		
2 EXHAUSTIEF ZOEKEN	sectie				
Alle mogelijkheden afgaan	opdracht	+	+		
Aantal mogelijke reiswegen (4x4m)	pagina		+		+
Aantal mogelijke reiswegen (algemeen)	opdracht	+	+		+
Problemen met exhaustief zoeken	opdracht		+	+	+
3 GULZIG ZOEKEN	sectie				
Het principe van gulzig zoeken	pagina		+		+
Voer het zoekalgoritme uit op enkele voorbeelden*	test	+	+		
Evaluatie gulzig zoeken	test			+	+
Lokale maxima	pagina		+		+
Programmeeropdracht*	test	+	+		
4 DYNAMISCH PROGRAMMEREN	sectie				
Een derde algoritme	pagina			+	+
Een vergelijking (delen 1-3)	pagina		+		+
Een vergelijking (deel 4)	pagina	+	+		
Een vergelijking (deel 5)	opdracht	+	+		
De maxmonstertabel	pagina			+	+
Stel de *mam* op voor enkele marslandschappen*	test	+	+		
Programmeeropdracht: maxmonstertabel	test	+	+		
5 COMPUTATIONEEL DENKEN	sectie				
Elementen computationeel denken	feedback			+	
6 UITBREIDINGEN	sectie				
Uitbreidingen	pagina	+	+		
Variaties	pagina	+	+		
7 EINDE	sectie				
Afsluiter	pagina				+

* Afgekorte naam van de activiteit

Tabel 4.4: Het standaard leerpad van les marslander met de eigenschappen van de verschillende activiteiten en snelkoppelingen naar de verschillende secties.

4.5.3 Elementen computationeel denken

In deze les komen vooral *decompositie*, *algoritmisch denken* en *evaluatie* aan bod. Er wordt ook aan *abstractie* gedaan, maar dat is niet de focus van deze les. De volledige bespreking van de kernelementen van CD in deze les is online terug te vinden[9].

Decompositie

Het dynamisch programmeren is een schoolvoorbeeld van *decompositie* van het probleem in deelproblemen van dezelfde aard. Het grote probleem wordt namelijk opgedeeld in steeds kleiner wordende deelproblemen (marslandschappen) tot men in een basisgeval terechtkomt. Daarna worden de oplossingen van de deelproblemen aangewend om het volledige probleem om te lossen.

Algoritmisch denken

Deze les zoomt in op drie verschillende algoritmes die kunnen worden gebruikt om het probleem aan te pakken. Twee daarvan, het gulzig zoekalgoritme en de dynamisch geprogrammeerde oplossing, worden door de leerlingen manueel uitgetest op enkele voorbeelden. Daarna kunnen ze hun oplossingen vergelijken met de modeloplossingen. Deze twee algoritmes worden ook geïmplementeerd in Python. Het gulzig zoekalgoritme blijkt hierbij ook zinvol bij het zoeken naar de beste reisweg in het dynamisch-geprogrammeerde algoritme.

Evaluatie

In deze les is veel aandacht voor de voor- en nadelen van de verschillende algoritmes. Zowel de snelheid, correctheid als de moeilijkheidsgraad om deze te programmeren, worden belicht. Bij het gulzig zoekalgoritme wordt ook uitgebreid ingegaan op de complexiteit die komt kijken bij marslandschappen van willekeurige grootte.

4.5.4 Ontwerpkeuzes

Idee van deze les

Het idee achter deze les is bewust zo gekozen dat aspecten en elementen van CD die nog niet aan bod kwamen in eerder uitgewerkte lessen, in deze les verwerkt kunnen worden. Zo wordt er, in tegenstelling tot de drie andere lessen, niet gewerkt met grafen in deze les. Niet elk probleem kan namelijk worden omgezet in een graaf. Daarnaast focust het dynamisch programmeren op *decompositie*, het element van CD dat nog het minst aan bod kwam in eerder uitgewerkte lessen.

[9] `$code/mod/feedback/view.php?id=197`.

Langere les
Deze les is uitgegroeid tot een langere les dan initieel werd beoogd. Dit is ook te merken aan de verschillen tussen de mock-up (appendix C) en de uiteindelijke les. Aangezien de toegevoegde elementen wel degelijk een meerwaarde bieden, werd de les een omvangrijk geheel. Zoals werd beschreven in 4.2.4 kan de leerkracht zelf heel wat keuzes maken om de les aan te passen, uit te breiden of in te korten.

Uitgebreide inleiding
Deze les start met een uitgebreide inleiding. Daarin komen verschillende animaties voor met voorbeelden van de probleemstelling (zie 3.5.2.2). Op die manier raakt de leerling vertrouwd met het probleem van de marslander. De gerichte feedback die deze animaties geven, zorgen dat leerlingen meestal een oplossing vinden.

Drie algoritmes
Er worden doorheen de les drie verschillende algoritmes bestudeerd in een welbepaalde volgorde. Hierbij worden heel wat voorbeelden en ondersteunende beeldmaterialen gebruikt. Het is mogelijk om bepaalde algoritmes over te slaan en/of de volgorde te veranderen.

Het exhaustief zoekalgoritme is eenvoudig in die zin dat het idee erachter gemakkelijk te begrijpen is en leerlingen zelf kunnen inzien dat het steeds een optimale oplossing vindt. Ontrafelen hoe men alle routes moet overlopen en hoeveel dat er zijn is heel wat minder eenvoudig. Dit algoritme implementeren is ook geen sinecure en komt daarom niet aan bod in deze les. Uiteraard is het mogelijk om dit toe te voegen of te behandelen buiten Co-De.

Vertrekkende van de nadelen van het eerste algoritme wordt een gulzig zoekalgoritme opgesteld. Dit algoritme is vrij eenvoudig te begrijpen en te implementeren. In dit gedeelte wordt dus wel geprogrammeerd. Aan de hand van een aantal voorbeelden wordt toegelicht dat dit algoritme niet altijd de beste route vindt.

Het dynamisch programmeren volgt bewust pas als laatste algoritme. We vertrekken hiervoor vanuit het idee om de nadelen van beide voorgaande algoritmes weg te werken. Bovendien kan men via de *maxmonstertabel* de beste route vinden met de gulzige zoekstrategie die reeds werd toegelicht. Het algoritme wordt ook geprogrammeerd in twee stappen. De implementatie is niet lang, maar vereist wel wat denkwerk.

Aantal mogelijke routes
Gespreid over een aantal pagina's wordt uitgelegd hoeveel routes er bestaan in een bepaald marslandschap. Hierbij worden veel illustraties gebruikt en het proces is opgedeeld in verschillende bevattelijke deelstappen.

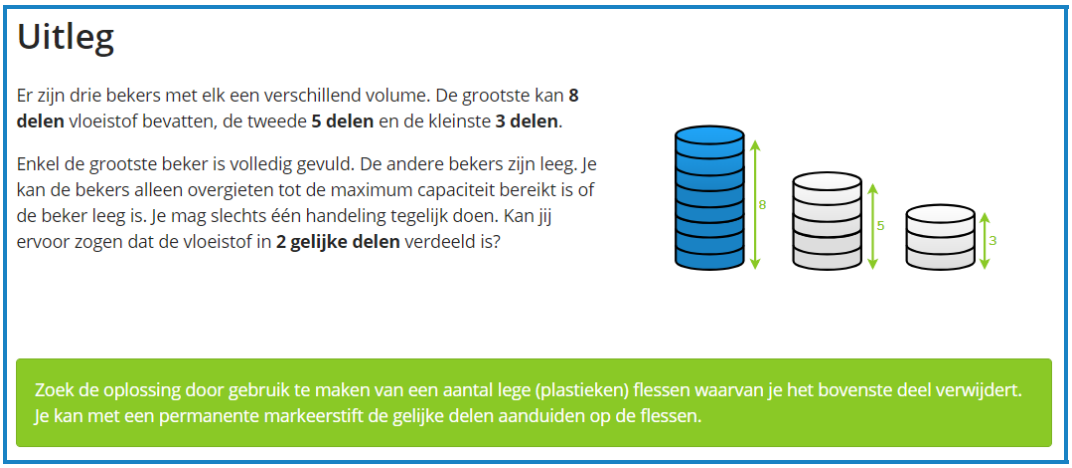
Er komt ook een formule uit kansrekenen aan bod. Zo wordt exact vastgelegd hoeveel routes er zijn voor (vierkante) marslandschappen van willekeurige groottes. Op deze manier kan ook de link worden gelegd met de lessen wiskunde. Zoals wordt beschreven in 4.2.4 kan dit onderdeel worden overgeslagen.

4.6 Les 4: de drie bekers

De demo van de deze les is beschikbaar via `$code/course/view.php?id=11`.

4.6.1 Probleemstelling

De drie bekers gaat over een puzzel met enkele eenvoudige instructies (zie figuur 4.10). De bekers hebben elk een verschillend volume. De grootste beker is volledig gevuld met een vloeistof en op elk moment mag de inhoud van een beker overgegoten worden in een andere beker. Het doel is om op het einde de vloeistof in twee gelijke volumes verdeeld te hebben. Je kan enkel blijven gieten tot de andere beker vol is (er blijft nog een hoeveelheid over in de oorspronkelijke beker) of tot de beker die je overgiet leeg is. Het is niet mogelijk om twee bekers tegelijkertijd over te gieten. De opdracht wordt eerst *unplugged* uitgevoerd: leerlingen mogen met echte bekers proberen een oplossing te vinden.



Figuur 4.10: De hoofdopdracht uit de drie bekers

4.6.2 Standaard leerpad

In het standaard leerpad (zie tabel 4.5) krijgen de leerlingen de puzzel met drie bekers te zien. Ze moeten de tijd krijgen om hiervoor een oplossing te zoeken en die op een (leuke, duidelijke of verrassende) manier te presenteren. Het formaat hiervoor ligt niet vast; de oplossing wordt ingestuurd als een opdracht en mag multimediabestanden bevatten.

Daarna worden er een aantal vragen gesteld over hun inzending zoals “Hoeveel oplossingen hebben ze gevonden?”, “Welke technieken hebben ze gebruikt?” en “Hoeveel oplossingen zijn er maximaal?”. Dit onderdeel heeft als doel hen de opgave nog wat verder te laten analyseren.

De derde sectie toont een exacte oplossingsmethode voor de puzzel. Die methode houdt in om eerst eens naar een toestandstabel te kijken en nadien een gerichte graaf op te stellen van de mogelijke overgangen tussen de toestanden. Verder worden een aantal mogelijke oplossingen van de puzzel besproken.

In het volgende deel komen een aantal gelijkaardige puzzels aan de bod. Voor deze puzzels worden geen oplossingen aangeboden in de les zelf. Wel beschikt de leerkracht in het deel "informatie voor leerkrachten" over de bronnen van de puzzels en extra informatie.

Tot slot volgt de standaard vragenlijst over de elementen van computationeel denken. In deze vorm neemt de les ongeveer **één lesuur** in beslag.

DE DRIE BEKERS		Doe	Denk	Reflecteer	Illustreer
1 INLEIDING	sectie				
Uitleg	pagina	+	+		
2 OPDRACHT	sectie				
Oplossing voor de drie bekers	opdracht	+	+		
Hoeveel oplossingen?	keuze			+	
Reflecteer	test	+	+	+	
Welke techniek heb je gebruikt?	keuze			+	
3 GRAAFVOORSTELLING	sectie				
Graafvoorstelling: uitleg	pagina	+	+		+
Graafvoorstelling: uitwerking	pagina				+
Graafvoorstelling: oplossingen	pagina				+
4 COMPUTATIONEEL DENKEN	sectie				
Elementen computationeel denken	feedback			+	
5 VARIATIES & UITBREIDINGEN	sectie				
Variaties	pagina	+	+		
Variatie: De Drie Kopjes	pagina	+	+		
Variatie: De Wolf, de Geit en de Kool	pagina	+	+		
6 EINDE	sectie				
Afsluiter	pagina				+

Tabel 4.5: Het standaard leerpad van de les de drie bekers met de eigenschappen van de verschillende activiteiten en snelkoppelingen naar de verschillende secties.

4.6.3 Elementen computationeel denken

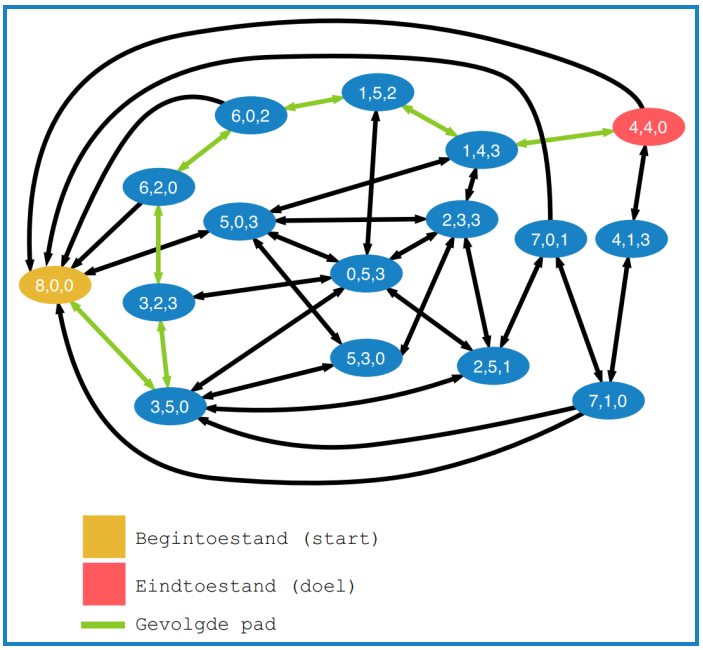
In deze les komen voornamelijk de elementen *abstractie* en *veralgemening* van computationeel denken aan bod. De volledige bespreking van de kernelementen van CD in deze les is online terug te vinden[10].

Abstractie

Abstractie komt voornamelijk aan bod wanneer leerlingen een formele oplossing uitwerken voor de puzzel. Ze leren hoe een volledige toestandstabel wordt opgesteld. Vooreerst wordt een notatiewijze aangereikt om een toestand van de drie bekers voor te stellen, namelijk aan de hand van een tripel (figuur 4.12). Verder wordt een formele notatie uitgewerkt die de handeling "Giet een beker over in een andere beker" exact weergeeft (zie figuur 4.12). Nadien wordt er nog meer geabstraheerd: toestandsovergangen waarvan de eindtoestand dezelfde is als de starttoestand mogen achterwege gelaten worden. Dit leidt uiteindelijk tot een gerichte graaf waarin het voldoende is een pad te vinden van de start- naar de doelknoop (figuur 4.11).

Veralgemening

Om leerlingen vertrouwd te maken met het omzetten van gegeven puzzels naar grafen worden er gelijkaardige puzzels aangeboden in het onderdeel variaties & uitbreidingen. Concreet zijn er twee puzzels toegevoegd: "De Drie Kopjes" en "De Wolf, de Geit en de Kool". Daarnaast zijn er nog een aantal variaties op de initiële puzzel, zoals werken met andere hoeveelheden en bekers.



Figuur 4.11: De gerichte graaf voor de drie bekers met markering van één van de mogelijke paden.

[10] $code/mod/feedback/view.php?id=184.

Figuur 4.12: De toestandstabel met alle overgangen voor de puzzel uit de drie bekers. Één van de mogelijke oplossingen is aangeduid in het groen.

.

4.6.4 Ontwerpkeuzes

Lesduur
In vergelijking met de andere drie lessen is de drie bekers beduidend minder groot in omvang. Dit maakt de les geschikt om ook op korte momenten in te zetten en als tussendoortje te behandelen. Het nodigt de leerkracht uit om eigen ideeën toe te voegen aan de lesspecifieke inhoud of buiten Co-De te werken rond een soortgelijk onderwerp.

Keuze van de activiteiten
Voor deze puzzel werd er geen animatie uitgewerkt zoals bij andere lessen, maar wordt er gevraagd om de oefening eerst *unplugged* op te lossen. In het verdere verloop van de les worden de concepten (gerichte grafen, toestandsautomaten...) telkens geïllustreerd aan de hand van een tekstuele uitleg en duidelijke afbeeldingen. Het is op die manier mogelijk om de les volledig zelfstandig te volgen, bijvoorbeeld wanneer een leerling al klaar is met andere opdrachten.

Variaties
Omdat de les een relatief kleine omvang heeft, zijn er minder mogelijkheden om het leerpad te laten variëren. De grootste vrijheid ligt in het wel of niet volgen van de variaties op de puzzel aan het einde van de les. De variaties zijn vooral belangrijk om de leerlingen te trainen in *veralgemening*. De variaties zijn namelijk allemaal puzzels die kunnen getransformeerd worden tot een gerichte graaf of toestandsautomaat, waaruit dan kan afgelezen worden of er al dan niet een oplossing is voor de puzzel.

Vragenlijst "elementen computationeel denken"
De standaard vragenlijst over de elementen van CD is in deze les bewust meer aan het einde geplaatst. Ze staat hier vlak voor de afsluiter in plaats van voor de variaties & uitbreidingen. Dit is nodig omdat *veralgemening* vooral aan de beurt komt in de gelijkaardige puzzels uit de variaties.

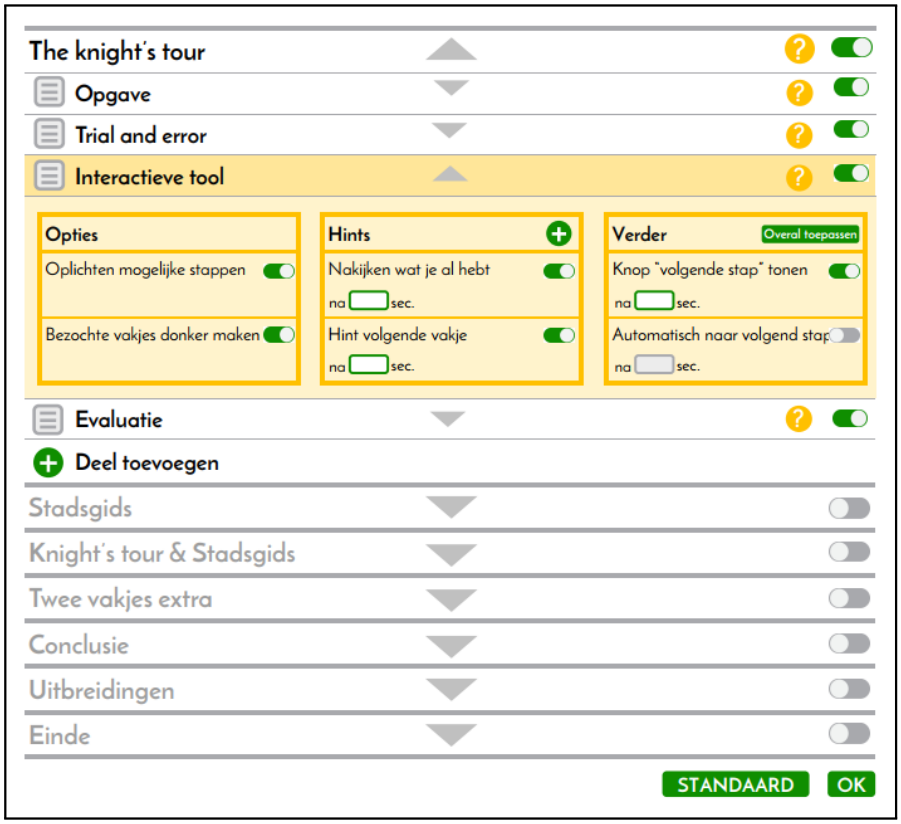
4.7 Besluit

Dit hoofdstuk bespreekt de verschillende uitgewerkte lessen op Co-De. Dit gebeurt aan de hand van de beknopte probleemstelling, het standaard leerpad en de specifieke ontwerpkeuzes die per les worden gemaakt. De vier lessen vormen een gebalanceerd geheel, zowel qua verdeling van de kernelementen van computationeel denken als in hun opbouw en lesduur. Om de opbouw van elke les te analyseren, wordt gebruikgemaakt van het 4C/ID-model en vier eigenschappen die worden toegekend aan de lesactiviteiten: *doe*, *denk*, *reflecteer* en *illustreer*. Om de ontwikkeling van nieuwe lessen in de toekomst te faciliteren, wordt ook het ontwerpproces van een les uitvoerig besproken.

Hoofdstuk 5

Nabeschouwing

In deze nabeschouwing wordt teruggeblikt op het afgelopen jaar en wordt vooruitgekeken naar de toekomst van Co-De. Eerst worden een aantal aspecten besproken die de toekomst van het leerplatform mee zullen bepalen. Daarna worden enkele moeilijkheden toegelicht waar we het afgelopen jaar mee te maken hebben gekregen. Ten derde worden een aantal ideeën opgesomd in verband met verder werk waar binnen het kader van deze masterproef geen tijd meer voor was. Tot slot volgt een kort besluit.



Figuur 5.1: Een mock-up van de gebruikersinterface voor de leerkrachten zoals die er in een ideaal scenario zou uitzien. Deze mock-up werd gemaakt om de vereisten te analyseren nog voor de keuze voor Moodle gemaakt werd.

5.1 Co-De in de toekomst

Co-De is opgestart in het academiejaar 2017-2018 en online beschikbaar sinds februari 2018. Er is heel wat afgewerkt en bruikbaar materiaal op het platform beschikbaar en iedereen kan zich gratis registreren. Het team achter Co-De ziet het afronden van deze masterproef niet als het eindpunt voor het leerplatform. Daarom worden hier enkele aspecten besproken die de toekomst van Co-De mee zullen bepalen.

5.1.1 Actoren

De verschillende partijen die betrokken zijn bij Co-De geven aan hun engagement te willen verderzetten. Zo zal de hosting en de technische infrastructuur die ter beschikking werd gesteld door de KU Leuven (zie 3.4) operationeel gehouden worden door het departement computerwetenschappen.

De promotoren achter deze masterproef, prof. dr. Bart Demoen en prof. dr. Bern Martens, blijven ook betrokken bij Co-De. Zij willen een actief gebruik ervan mee promoten en zelf Co-De gebruiken waar mogelijk. Daarnaast trachten zij om nieuwe masterstudenten aan te trekken om, in het kader van hun masterproef, Co-De verder uit te bouwen. Hieronder valt zowel Co-De onderhouden als nieuwe functionaliteiten en lessen toevoegen.

Wij, de auteurs van deze tekst, zijn ook bereid om het platform in stand te blijven houden en ondersteuning te bieden bij nieuwe initiatieven rond Co-De. We dragen het hele project immers een warm hart toe en hopen dat het in de toekomst ook in de praktijk kan worden ingezet.

5.1.2 Onderhoud

Op technisch vlak vergt het platform regelmatig onderhoud en updates. Zo zijn er frequent updates beschikbaar van Moodle en van de gebruikte plugins. Soms zijn deze optioneel, maar af en toe bevatten deze ook veiligheidsupdates die men beter niet te lang uitstelt. Daarnaast moeten ook het platform zelf, de Docker-containers, de mailserver en het `SSL`-certificaat onderhouden worden. Deze aspecten worden in detail besproken in hoofdstuk 3 en werden zoveel mogelijk geautomatiseerd.

De hoeveelheid onderhoud die nodig is hangt, tot op zekere hoogte, samen met hoeveel Co-De gebruikt zal worden. Indien Co-De niet zoveel wordt gebruikt door scholen, is het minder cruciaal om kort op de bal te spelen met updates. Indien het gebruik sterk verhoogt, is dit wel nodig en is het ook belangrijk om de serverinfrastructuur in de gaten te houden. Deze is namelijk enkel getest voor een belasting die een klas simuleert.

In de verdere toekomst is het ook belangrijk om de opgeslagen data en back-ups te beheren. Hetzelfde geldt voor de mailserver(s) en gebruikersaccounts.

5.1.3 Gebruik

Co-De is gebruiksklaar en werd al gebruikt door een klas (zie 3.7 vanaf p. 66). De verschillende (technische) aspecten van het platform zijn individueel getest en een uitgebreide gebruikershandleiding is online beschikbaar voor iedere gebruiker[1].

Op deze manier hopen we scholen en leerkrachten warm te kunnen maken en voldoende ondersteuning te bieden om Co-De te gaan gebruiken. In dit kader werd het leerplatform ook reeds voorgesteld aan een 40-tal leerkrachten op het slotevent van progra-MEER 2017-2018[2].

Om een actieve gebruikersgroep op te bouwen is het nodig om in de toekomst proactief te werken met testgebruikers en klassen. Nu het platform werkt en online beschikbaar is, is het ook eenvoudiger om nieuwe lessen en functionaliteiten te laten uittesten door klassen en scholen. Dit was niet mogelijk in de opstartfase waarin het platform en de lessen tegelijk ontwikkeld werden.

5.1.4 Bijdrage WiPSCE

WiPSCE staat voor "Workshop in Primary and Secondary Computing Education". Het is een conferentie die in oktober 2018 plaatsvindt te Potsdam in Duitsland. Het thema van de conferentie is het lager en secundair informatica-onderwijs en de daaraan verbonden lerarenopleidingen en onderzoeksdomeinen [36]. WiPSCE heeft als doel de internationale uitwisseling van onderzoek en initiatieven in dat veld te bevorderen [36].

In juni 2018 wordt een (Engelstalige) posterpaper over Co-De ingestuurd voor WiPSCE. Op deze manier kan Co-De ook op internationaal vlak aan bekendheid winnen en kunnen zinvolle ideeën worden uitgewisseld. Daarnaast wordt op deze manier ook het onderzoeksaspect van dit project benadrukt dat in het gebruik soms minder aan bod komt. Tot slot geeft deze bijdrage aan WiPSCE ook aanleiding tot het vertalen van het platform en de lesinhoud naar het Engels zoals verderop wordt beschreven in 5.3.3.

[1] `$code/handleiding`.
[2] Website progra-MEER: `http://www.progra-meer.org/`.
Slotevent 2017-2018: `https://www.uhasselt.be/progra_meer_slotevent`.
Video-opname van de presentatie: `https://www.youtube.com/watch?v=5GhX6uMRKTU`.

5.2 Moeilijkheden

Tijdens het onderzoek waren er een aantal obstakels die moesten worden overwonnen. Er volgt een kort overzicht van de belangrijkste hindernissen.

5.2.1 Literatuur

Het zoeken en raadplegen van geschikte literatuur was de eerste grote uitdaging binnen het onderzoeksproject van Co-De. De literatuur over computationeel denken is enorm uitgebreid. Ook het aantal onderwijsprojecten rond dit begrip is de laatste jaren enorm toegenomen. Aan het begin van onze masterproef was nog niet duidelijk welke informatie daarvan nodig zou zijn binnen ons project.

Specifieke literatuur voor softwareontwikkelaars over het ontwerpen van leerplatformen is er niet. De meeste platformen worden op de markt gebracht als commercieel product. Bijgevolg is er voor buitenstaanders weinig informatie over de opbouw ervan. Andere applicaties zijn vaak al verouderd en vormen dus geen goede basis op het vlak van nieuwe onderwijstechnologie. *Open source* initiatieven boden een veel betere inkijk en waren nagenoeg de enige informatiebron. Voor zulke initiatieven is duidelijke en uitgebreide documentatie noodzakelijk. Publicaties over software-architectuur zijn meestal voor een algemene context bedoeld. Er komen weinig structuren voor leertoepassingen in aan bod.

De literatuur over de situatie van het (informatica)onderwijs in Vlaanderen en in de buurlanden schetste een goed, maar soms troebel, kader voor deze masterproef. Dit had echter uiteindelijk weinig rechtstreekse invloed op het leerplatform zelf. Terugblikkend had de literatuurstudie over het onderwijs korter mogen zijn geweest. Vandaar dat deze literatuur ook slechts vluchtig aan bod komt in hoofdstuk 2 over computationeel denken.

5.2.2 Inhoudelijk

Op inhoudelijk vlak waren er een aantal overwegingen die grote consequenties met zich meebrachten. Er waren tijdens het project ongeveer 780 man-uren (60% van de projectomvang) voorzien om het platform en de lesinhoud vorm te geven. Aan het begin van het project werd beoogd om in deze tijdspanne een volledige lessenreeks te ontwikkelen (voor één wekelijks lesuur tijdens één schooljaar). Dit bleek in de praktijk echter moeilijk haalbaar. Als er meer lessen uitgewerkt zouden worden, dan had de functionaliteit van het platform daaronder geleden. Er werd uiteindelijk ook gekozen om de vier ontwikkelde lessen zo volledig mogelijk af te werken in plaats van de andere onderwerpen (zie 4.2.2.1) nog volledig of gedeeltelijk uit te werken. Zelfs na de beperking tot vier complete lessen, zijn de mogelijkheden op het platform soms gelimiteerd. Zo werd het aanvragen van *eigen lessen* op Co-De nog niet geautomatiseerd. Door toekomstig gebruik van het platform zal hopelijk blijken welke inhoud zal moeten worden bijgestuurd.

5.2.3 Het leerplatform

Het ontwikkelen van het platform kende zijn eigen moeilijkheden. Zoals aangegeven in hoofdstuk 3, was aan het begin van het project niet duidelijk of er zou gewerkt worden met een bestaande applicatie of dat er een eigen platform zou worden gebouwd. Het was niet evident om te beslissen welke functionaliteiten beter wel of niet ondersteund moesten worden. Als masterstudenten hadden we beiden weinig ervaring in het bouwen van grotere applicaties. De planning zou daarom misschien niet altijd even nauwkeurig kunnen worden opgevolgd. Uiteindelijk werd half november 2017 beslist om met Moodle van start te gaan. Daarna volgde eerst een proefperiode om uit te zoeken of Moodle flexibel genoeg was om de lesideeën te presenteren zoals uitwerkt in de mock-ups.

Aanvankelijk werd met een lokale standaardinstallatie van Moodle gewerkt. Nadien moest de applicatie naar de *cloud* verplaatst worden. Gezien Co-De potentieel heel wat simultane gebruikers moet kunnen ondersteunen, was het zoeken naar een gepaste server[3]. Er waren zeker mogelijkheden voor kant-en-klare Moodle-hosting, maar de prijzen hiervoor lopen erg op wanneer de applicatie moet worden geschaald. Het aanpassen van Moodle is eenvoudiger wanneer volledige servertoegang mogelijk is. Vanuit het departement computerwetenschappen kwam de suggestie om Co-De op te zetten met Docker. Ook dit was een groot leerproces, waarbij het niet meteen duidelijk was hoe Moodle op een duurzame manier kon worden opgebouwd. Om het onderhoud aan Co-De zo klein mogelijk te houden, moesten veel repetitieve taken geautomatiseerd worden. Hier werd een aanzienlijk deel van onze tijd aan besteed. De automatisatie draagt niet onmiddellijk bij tot een beter (visueel) resultaat of een meer gestroomlijnde gebruikerservaring, maar is wel van belang voor het project op lange termijn.

Ten slotte rezen in de loop van het onderzoeksproject nog andere implementatiegerelateerde vragen. Zo werd bijvoorbeeld onderzocht of Co-De onmiddellijk de nieuwe regulering betreffende persoonsgegevens zou kunnen volgen (zie 5.3.5). Een ander voorbeeld is het terugzetten van een *eigen les* naar de standaardinstellingen van de *demo-les*. Dit is een handeling die door leerkrachten gewenst zou kunnen zijn, maar nog niet op een gebruiksvriendelijke manier mogelijk is in Co-De.

5.2.4 Vakdidactiek

Wij hadden, als masterstudenten computerwetenschappen, weinig (vak)didactische kennis bij aanvang van deze masterproef: (vak)didactiek maakt immers geen deel uit van onze opleiding. Binnen de context van dit onderzoek was het eveneens niet mogelijk om deze ervaring op te doen. Dit maakte het moeilijk om het doelpubliek van Co-De goed in te schatten en de lesinhoud daarop af te stemmen. Regelmatig moesten ideeën of ontwerpen daarom worden afgetoetst bij de begeleiders van deze masterproef.

[3] https://docs.moodle.org/34/en/Installing_Moodle#Requirements.

5.3 Verder werk

5.3.1 Optimalisatie gebruik

Het leerplatform is volledig operationeel en de mogelijkheden zijn uitgebreid gedocumenteerd in de gebruikershandleiding[4]. Desalniettemin zijn sommige functionaliteiten nog niet optimaal uitgewerkt om ze als gebruiker te hanteren (zie bijvoorbeeld figuur 5.1, vooraan dit hoofdstuk, van een functionaliteit die nog niet volledig werd gerealiseerd). Hieronder volgen drie belangrijke functionaliteiten waar nog verbetering mogelijk is, maar dit zijn zeker niet de enige gevallen.

Ten eerste moeten *eigen lessen* manueel worden aangemaakt door beheerders. Dit is een proces dat op termijn geautomatiseerd zou kunnen worden. Ook de structuur van de *eigen lessen* op Co-De kan beter. Iedereen kan namelijk ook de *eigen lessen* van andere scholen zien. Dat betekent niet dat iedereen die lessen ook kan gebruiken of de leerlingengegevens kan zien. Deze structuur zou moeten verfijnd worden, zodat ze ook voldoet aan de GDPR-regelgeving (zie 5.3.5).

Een tweede element dat zeker nog voor verbetering vatbaar is, is de opsplitsing van de rollen. Zoals wordt beschreven in 3.5.3 worden de meeste rollen toegewezen per les. Toch zou op Co-De een breder onderscheid moeten worden gemaakt tussen leerlingen en leerkrachten. Leerlingen zouden zich bijvoorbeeld niet als demo-leerkracht mogen kunnen inschrijven voor een *demo-les* of een les aanvragen, wat nu wel kan. Daarenboven zal op termijn ook meer nuance moeten worden aangebracht in de verschillende (extra) leerkrachtenrollen zoals reeds werd beschreven bij de toelichting van deze rol.

Het ontbreekt Co-De momenteel ook aan een toegankelijk portaal buiten het leerplatform zelf. Om potentiële nieuwe gebruikers goed te informeren is, op termijn, een portaalpagina of website vereist. Deze moet beschikbaar zijn zonder ingelogd te moeten zijn op het platform. Ook de handleiding moet beschikbaar worden gesteld zonder eerst te moeten inloggen op het platform. In dit verband wordt overwogen om een eigen (toegankelijker) webdomein aan te kopen, bijvoorbeeld `www.co-de.be`.

5.3.2 Uitbreiden van de lessen

Een ander belangrijk aspect om Co-De verder te doen groeien, is het uitbreiden van de lessenreeks. Het is daarbij gewenst om het lessenpakket samen te stellen als een evenwichtig geheel. Dit kan zowel op vlak van de kernelementen van CD (2.3) die getraind worden, als de samenstelling binnen de les van de vier soorten activiteiten (*doe*, *denk*, *reflecteer* en *illustreer*) (4.2.2.3). Verder is het ook van belang om het 4C/ID-model te blijven respecteren (4.2.1).

Enkele nieuwe onderwerpen bevinden zich al in de idee- of mock-upfase. Er is onder andere een mock-up uitgewerkt voor een les over sorteeralgoritmes. Over die mock-up

[4] `$code/handleiding`.

werd al inhoudelijk geïtereerd, maar er zijn nog wat aanpassingen nodig vooraleer de les op Co-De kan geplaatst worden. Daarnaast zijn er lesonderwerpen besproken waarvoor zeker ruimte is binnen het huidige lessenpakket. Zo zou een les rond encryptie uitgewerkt kunnen worden. Dit onderwerp is zeer actueel en jongeren hebben er maatschappelijk belang bij om op de hoogte te zijn van de veiligheid van softwaresystemen. Vervolgens biedt het leerplatform een goede basis om verder aan de slag te gaan met animaties om één of meerdere lessen rond simulaties uit te werken. Interessante voorbeelden van simulaties kunnen bekeken worden via "Netlogo"[5]. Ook schriftelijke bronnen met lessen rond CD kunnen in een herwerkte, interactieve vorm gepaste lessen zijn voor Co-De. Met name "Abenteuer Informatik"[6] is daar een voorbeeld van.

Aansluitend moet er ook ruimte vrijgehouden worden om de huidige lessen te evalueren en te herzien. Zo bleek bij de analyse van doolhoven in het vorige hoofdstuk (4.4), dat er nog *doe*- en *denk*-activiteiten moeten worden geplaatst voor de twee programmeeronderdelen. Anders worden er na de instructies van de zoekalgoritmes te weinig leertaken aangeboden waar die algoritmes kunnen worden ingeoefend.

5.3.3 Co-De in verschillende talen

De eerste stappen werden gezet om het platform in andere talen, zoals het Engels en Frans, beschikbaar te maken. Om het platform op grotere schaal uit te bouwen en Co-De voor te stellen aan internationaal publiek (zoals bijvoorbeeld op WiPSCE, zie 5.1.4) is een Engelstalige versie nodig. Enerzijds moeten een aantal platformafhankelijke elementen een vertaling krijgen. Dit is onder meer het geval voor de zelfgeschreven plugin "relatieve voltooiing", de aangepaste plugins (de voortgangsbalk en programmeeropdrachten) en het thema "code". Anderzijds moet de lesinhoud vertaald worden. Het schakelen tussen verschillende talen is al in Moodle (en dus ook in Co-De) aanwezig. Er moet enkel nog een systeem worden opgezet dat voor de lessen verschillende talen mogelijk maakt.

5.3.4 Logs analyseren

Tot slot is er ook nog heel wat onontgonnen informatie ter beschikking die kan gebruikt worden om Co-De verder uit te bouwen en te verbeteren. Co-De houdt immers logs bij van alle acties die worden uitgevoegd op het platform[7]. In Moodle wordt heel wat informatie gelogd zoals: wie betreedt welke les, hoelang wordt er stilgestaan bij een lesonderdeel, hoeveel keer werd er fout geantwoord op een vraag, hoe vaak werd welke term ingegeven in het cursus-zoekvenster...

Momenteel zijn er te weinig gebruikerslogs uit authentieke klassituaties beschikbaar om hier algemene conclusies uit te trekken. In de toekomst zal het zeker zinvol zijn om het gedrag van de gebruikers te analyseren om zo het platform en de lessen te verbeteren.

[5] https://ccl.northwestern.edu/netlogo/.
[6] Website horend bij het boek: http://www.abenteuer-informatik.de/index.html.
[7] https://docs.moodle.org/34/en/Logs.

5.3.5 GDPR

Vanaf mei 2018 trad de nieuwe "Algemene Verordening Gegevensbescherming" (AVG) in werking, deze is beter bekend onder de Engelse term "General Data Protection Regulation" (GDPR). Zoals de naam al aangeeft gaat deze over het beheer en de beveiliging van persoonlijke gegevens, meer bepaald van Europese burgers [8]. Deze regelgeving is van toepassing op alle bedrijven, organisaties, verenigingen en individuen die persoonsgegevens verwerken of opslaan [27]. Ze is dus ook belangrijk voor Co-De.

De GDPR bevat zowel regels met betrekking op technische aspecten van Co-De als op inhoudelijke elementen. Zo moeten databases voldoende afgeschermd zijn en moeten maatregelen genomen worden om de veiligheid te verzekeren. Daarnaast moeten de gebruikers ook duidelijk geïnformeerd worden welke gegevens voor hoe lang worden bijgehouden, wie daar toegang tot heeft en waarom dat nodig is. Elke gebruiker moet ook, op elk moment, de mogelijkheid hebben om zijn of haar gegevens integraal te laten verwijderen.

Binnen Moodle wordt hard gewerkt om de nodige functionaliteiten en aanpassingen betreffende deze regelgeving in te bouwen. Aangezien Moodle een *open source* platform is, vraagt dit enige tijd en overleg tussen de ontwikkelaars. Intussen zijn er hiervoor binnen Moodle al grote stappen gezet. Helaas kwamen deze te laat om nog ten volle in rekening te nemen voor Co-De[8]. Met het oog op het verder uitbouwen van Co-De moeten de GDPR-regels wel gevolgd worden. De belangrijkste informatie in verband met de GDPR op Moodle is terug te vinden via deze snelkoppelingen:
- https://docs.moodle.org/34/en/GDPR
- https://docs.moodle.org/34/en/GDPR_for_administrators
- https://moodle.com/news/moodle-gdpr-approach-plan/

5.3.6 Verdere automatisatie van de backend

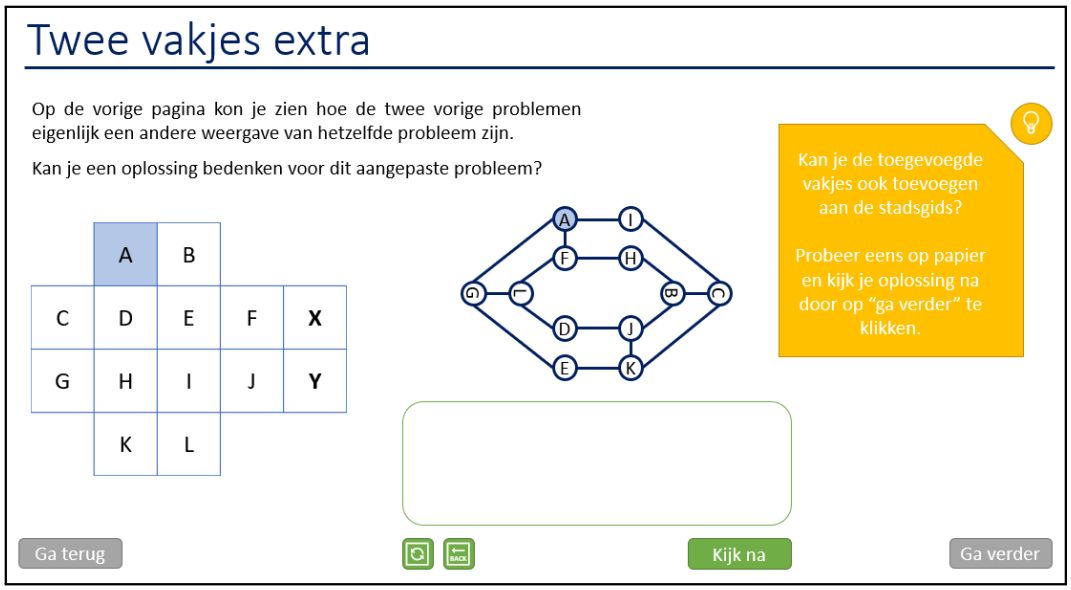
In de backend van Co-De kunnen nog meer taken geautomatiseerd worden. Het nieuwe SSL-certificaat dat tweemaandelijks geconfigureerd moet worden, is momenteel een manuele opdracht. Dit werd nog niet geautomatiseerd, omdat de Moodle-container in Docker voor deze taak herstart moet worden (zie appendix A). Het systeem is dan tijdelijk onbeschikbaar. Om er zeker van te zijn dat dit de eerste malen feilloos verliep, was toezicht nodig.

De lessen op Co-De worden dagelijks automatisch geback-upt. De back-up-bestanden worden echter op dezelfde machine bewaard als die waarop Co-De staat. Om gegevensverlies te voorkomen, zouden de back-up-bestanden beter gekopieerd worden naar een secundaire machine.

[8] Concrete richtlijnen werden gepubliceerd en via mail verspreid op 16 mei 2018.

5.3.7 Aanvullende functionaliteiten op Co-De ontwikkelen

Gedurende het onderzoeksproject zijn nog ideeën ontstaan voor extra functionaliteiten. De beperkte tijdspanne had echter tot gevolg dat deze elementen het daglicht nog niet zagen. Een dergelijk idee is om een gepersonaliseerd, interactief hint-systeem te bouwen. Deze hints komen dan (na een bepaalde tijd of aantal pogingen) tevoorschijn bij de lesonderdelen zoals wordt geïllustreerd op figuur 5.2. Dit idee sluit nauw aan bij het concept van procedurele informatie uit het 4C/ID-model (zie 4.2.1.1). Hierbij is het immers de bedoeling om deze informatie enkel indien het nodig is en "Just in time" (JIT) te geven [33].



Figuur 5.2: Een voorbeeld van een hint (afgebeeld in het geel) in de mock-up van de les paardenronde & stadsgids.

De plugin "relatieve voltooiing" maakt op dit moment nog geen deel uit van de standaard Moodle-pluginbibliotheek[9]. Om de plugin toe te kunnen voegen aan die bibliotheek moet er een heel proces doorlopen worden. De plugin moet ingestuurd worden en aanvaard worden door de beheerders van Moodle. Vooraleer de plugin aanvaard wordt, moet die aan een reeks richtlijnen voldoen. Zo moet de broncode op de juiste manier geformatteerd zijn en moeten er ook een aantal testen geschreven zijn (om de functionaliteit aan te tonen). Het formatteren van de broncode is al gebeurd. Voor de *testcases* werd een aanzet gedaan, maar dit zal in de toekomst nog meer tijd vragen. De Engelse vertaling van de plugin is al gerealiseerd. De volledige richtlijnen over het toevoegen van een plugin aan de Moodle-pluginbibliotheek zijn terug te vinden op de volgende URL: `https://docs.moodle.org/dev/Plugin_contribution`.

[9] `https://moodle.org/plugins/`.

5.4 Besluit

Na de succesvolle opstart van Co-De zijn er nog veel aspecten die in de toekomst aandacht zullen vragen. Ten eerste is er het regelmatig onderhoud dat het leerplatform vereist om operationeel te kunnen blijven. Ten tweede zijn er initiatieven nodig om het gebruik van het platform aan te moedigen, waaronder een nieuw masterproefproject en een bijdrage aan WiPSCE.

Tijdens het onderzoek waren er ook moeilijkheden waardoor de vooruitgang tijdelijk stagneerde. Op het vlak van literatuur was en blijft het moeilijk om geschikte bronnen te vinden. Voor de uitbouw van de lessen en het platform is vooral de tijd een beperkende factor. Daarnaast moesten doordachte technische keuzes worden gemaakt. De beperkte vakdidactische achtergrond van de auteurs heeft tot gevolg dat er frequent beroep moest worden gedaan op andere deskundigen.

Uiteindelijk werden in dit hoofdstuk nog een aantal elementen opgesomd die op termijn deel kunnen uitmaken van Co-De. Moodle is nog niet volledig in regel met de GDPR, maar heeft hier wel een werkmethode voor gepubliceerd. Het gebruik van Co-De kan nog meer gestroomlijnd worden door rollen gedetailleerder te definiëren en extra zaken te optimaliseren voor de gebruiker. Het aanbod aan lessen op Co-De is beperkt en zou in de toekomst zeker uitgebreid mogen worden. De lessen kunnen dan ook in verschillende talen beschikbaar worden gemaakt. In de backend zijn er nog handelingen of taken die geautomatiseerd moeten worden. Voor de applicatie zelf zijn er extra functionaliteiten bedacht die nog niet geïmplementeerd zijn, zoals het embedden van hints in de activiteiten. Ten slotte beschikt Moodle ook over gebruikerslogs die het mogelijk maken om het gebruik van Co-De te analyseren en daarna de resultaten van die analyse te gebruiken om het platform te verbeteren.

Hoofdstuk 6

Algemeen besluit

Het doel van deze masterproef was het opzetten van een digitaal leerplatform voor computationeel denken voor het secundair onderwijs. Dankzij dit leerplatform zouden leerlingen en leerkrachten toegang hebben tot een volledige lessenreeks over computationeel denken. De lessenreeks zou voldoende materiaal bevatten om één wekelijks lesuur informatica gedurende een gans schooljaar te geven op Co-De. Een bijkomende vereiste was dat de leerkracht de lessen op een gebruiksvriendelijke manier naar zijn hand zou kunnen zetten. De inhoud zou bij voorkeur op een multimediale manier worden gepresenteerd. Binnen de lessenreeks zou daarom ruimte moeten zijn voor zowel tekst en uitleg, *unplugged* oefeningen, online opdrachten, als voor programmeeropdrachten. Aan het platform ten slotte zou bijpassende functionaliteit kunnen worden toegevoegd dankzij een modulaire architectuur.

We hebben geopteerd voor een online leerplatform met de naam Co-De, verwijzend naar "Computationeel Denken". Sinds februari 2018 is Co-De online toegankelijk op `https://code.experiments.cs.kuleuven.be`. Op softwarematig vlak is het leerplatform gebaseerd op de leeromgeving Moodle om het klas- en lesmanagement te vergemakkelijken. De *open source* architectuur van Moodle garandeert dat we nieuwe functionaliteit konden invoegen (door middel van plugins). De server van Co-De bevindt zich aan het departement computerwetenschappen van de KU Leuven. De installatie van Co-De is daarop gerealiseerd door middel van Docker. Opdat het periodisch onderhoud van Co-De beperkt zou blijven, zijn zo veel mogelijk repetitieve taken uit de backend geautomatiseerd.

Om de lessenreeks te kunnen uitbouwen was een actuele en gefundeerde opvatting van computationeel denken (CD) noodzakelijk. In de literatuur zijn verschillende opvattingen over computationeel denken terug te vinden. Computationeel denken wordt daarbij als een verzameling van vaardigheden en attitudes beschouwd. De selectie van die vaardigheden is niet bij iedereen dezelfde waardoor de noties vaak overlappen, maar ze ook verschillen vertonen. In praktijk is er wel enige consensus en zijn de vaardigheden die het frequentst naar voren komen: *abstractie*, *veralgemening*, *decompositie*, *algoritmisch denken* en *evaluatie*. Op Co-De hebben we daarom deze vaardigheden verwerkt in ons uitgangspunt van CD.

In Vlaanderen staat de integratie van computationeel denken in het secundair onderwijs nog in de startblokken. Dankzij Co-De is er voor het eerst een integraal Nederlandstalig platform voor scholen dat lesmateriaal over CD aanbiedt. Op termijn kan de inhoud van Co-De ook in andere talen of formaten beschikbaar worden gemaakt.

De ontworpen lessenreeks bestaat momenteel uit vier lessen, het equivalent van ongeveer tien lesuren, waarbij elke les gecentreerd is rond een ander probleem. In elke les worden twee of drie van de kernelementen van CD prominent geoefend door het zoeken naar een oplossingsmethode voor het aangereikte probleem. De opbouw van een les wordt geïnspireerd op het 4C/ID-model en beoogt een gebalanceerde samenstelling van *doe-*, *denk-*, *reflecteer-* en *illustreer*-activiteiten. Op Co-De is de lessenreeks toegankelijk aan de hand van *demo-lessen*. Zo'n *demo-les* is geconstrueerd volgens de door ons bedachte lesinhoud (het standaard leerpad). Daarnaast kan de leerkracht verkiezen om te werken met *eigen lessen*. De *eigen lessen* vormen een exacte kopie van de bijbehorende *demo-les*, maar de leerkracht kan het leerpad aanpassen op maat van zijn klas. Dit kan door de activiteiten te herordenen, te wijzigen, te verbergen of nieuwe lesactiviteiten toe te voegen. Een overzichtelijke, maar uitgebreide handleiding[1] op maat van de leerkrachten en leerlingen brengt duidelijkheid over de mogelijkheden op Co-De. De eindgebruiker kan op die manier aan de slag met het platform zonder specifieke voorkennis over computationeel denken of ervaring met de Moodle-leeromgeving.

Er zijn ook enkele vooropgestelde doelen die, binnen deze masterproef, niet meer haalbaar bleken. Vier lessen zijn immers niet voldoende om één lesuur informatica per week gedurende één schooljaar te vullen. Hier was vooral de tijd een beperkende factor. De technische aspecten van het platform vroegen meer toewijding dan we hadden verwacht, maar ook de lessen zelf uitwerken bleek een tijdrovend proces. Het zoeken van goede en vernieuwende lesideeën is niet eenvoudig. Ook het vormgeven en opbouwen van een les vraagt heel wat exploratie en tijd. Het proces van lesonderwerp tot afgewerkte *demo-les* op Co-De gebeurt in verschillende ontwikkelingsfases. Tussen deze fases was er telkens nood aan overleg met de betrokken actoren. Bij toekomstig onderzoek is er dus zeker nog voldoende ruimte om nieuwe of gesuggereerde onderwerpen te verwerken in nieuwe lessen voor Co-De. Gebruikers en geïnteresseerden worden uitgenodigd om mee na te denken over nieuwe lessen[2]. Er zijn eveneens functionaliteiten die we aanvankelijk voor ogen hadden, maar die nog onbestaand of onvolledig zijn. Zo is er bijvoorbeeld nog geen hint-systeem in de opdrachten en is het nog niet mogelijk om *eigen lessen* volledig automatisch te creëren.

[1] `$code/handleiding`.

[2] Daarvoor is een forum beschikbaar op Co-De via `$code/forum`.

Het is belangrijk om aan te vullen dat we ook enkele zaken hebben gerealiseerd die niet in de oorspronkelijke doelstellingen vervat zaten. Ten eerste hebben we enkele testen gedaan (zie 3.7, p. 66) en hebben we de informatie uit deze testen ook aangewend om het platform te verbeteren. Ten tweede ontwikkelden we enkele plugins voor Moodle die ook andere Moodle-gebruikers kunnen installeren op hun eigen Moodle-site (zonder enig verband te houden met Co-De). Vooral de plugin "relatieve voltooiing" lijkt ons bijzonder nuttig voor andere Moodle-gebruikers. Een volgende stap in de ontwikkeling van die plugin is het klaarstomen van de code om de plugin in de plugin-bibliotheek van Moodle te plaatsen. Ten derde hebben we bijzonder veel aandacht besteed aan de eindgebruikers. Daar zijn de secties "informatie voor leerkrachten" en de uitgebreide gebruikershandleiding getuige van. Ten vierde werd Co-De een eerste maal voorgesteld aan het doelpubliek op het slotevenement van Progra-MEER op 24 april 2018 in Hasselt. Een 40-tal leerkrachten uit het secundair onderwijs en andere geïnteresseerden kregen daar een uiteenzetting over en demonstratie van het leerplatform te zien[3]. Tot slot hebben we, aan de hand van Moodle en Docker, een onverhoopt robuuste opstelling van ons platform kunnen realiseren met aandacht voor veiligheid, zelfregistratie en automatisatie van de backend.

Meer suggesties voor verder onderzoek zijn in detail beschreven in de nabeschouwing (hoofdstuk 5 vanaf p. 101). De belangrijkste taak die rest is het blijven uitbouwen van Co-De als platform voor computationeel denken en het lesmateriaal op peil houden, zodat geïnteresseerde scholen de weg vinden naar Co-De.

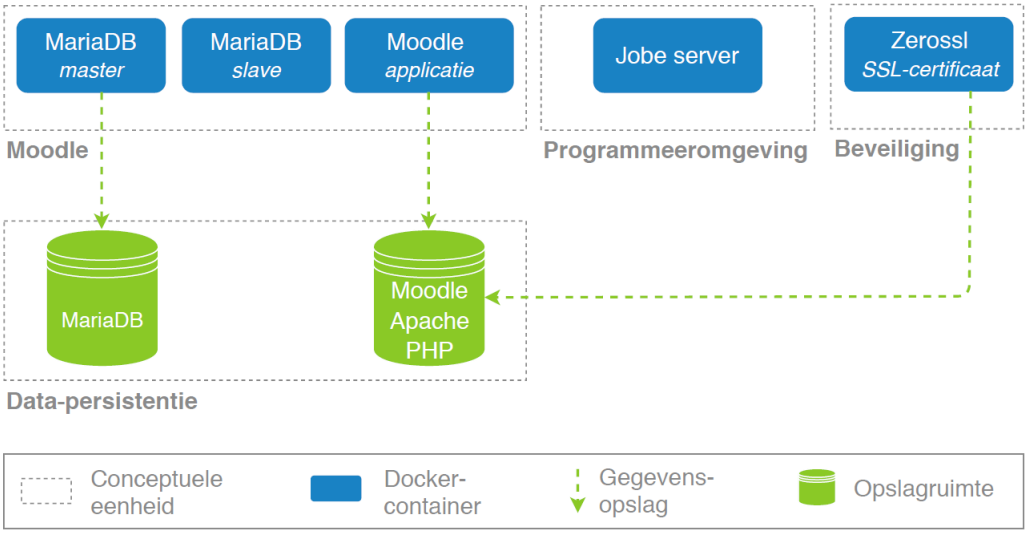
[3]Video-opname van de presentatie: https://www.youtube.com/watch?v=5GhX6uMRKTU.

Bijlagen

Bijlage A

Technische documentatie

Deze bijlage geldt als technische documentatie voor het opzetten van het leerplatform Co-De. Co-De is gebaseerd op de *open source* leeromgeving Moodle[1]. Moodle werd opgezet op een machine van het departement computerwetenschappen van de KU Leuven (`code.experiments.cs.kuleuven.be`). In dit document wordt vooreerst toegelicht hoe Moodle geïnstalleerd wordt en welk periodisch onderhoud vereist is. Daarnaast wordt dieper ingegaan op de "Jobe" server die nodig is voor programmeeropdrachten en de werking van de interactieve animaties die worden gemaakt in "Processing". Tot slot worden de wijzigingen aan Moodle die nodig zijn voor Co-De besproken. Hieronder vallen de plugins voor aanpassingen aan het thema en de functionaliteit van Moodle.



Figuur A.1: Het deployment van Co-De aan de hand van vijf Docker-containers. De data uit de database worden gepersisteerd op een schijfruimte. De data van de Moodle-applicatie worden tezamen met het SSL-certificaat gepersisteerd op een tweede schijfruimte.

[1] `https://download.moodle.org/`.

A.1 Docker-installatie van Moodle

In plaats van een traditionele installatie, werd Moodle voor dit project geïnstalleerd met Docker. Docker zorgt ervoor dat er geen aparte *dependencies* (PHP, MariaDB...) geïnstalleerd hoeven te worden. Docker is een platform dat het mogelijk maakt om programma's binnen geïsoleerde containers uit te voeren. Om een container op te starten, is een bijhorende *image*[2] van het gewenste programma noodzakelijk. Voor Co-De werd gebruik gemaakt van de Bitnami Moodle-*image*, die beschikbaar is op https://bitnami.com/stack/moodle.

De Bitnami Moodle-*image* vereist een tweede *image* om operationeel te zijn, namelijk een databasesysteem om de gegevens van de Moodle-applicatie bij te houden. Bitnami verwijst daarvoor naar de Bitnami Moodle-*image*, die beschikbaar is op https://bitnami.com/stack/mariadb.

A.1.1 Creatie van de containers

Hier worden de commando's opgesomd waarmee de originele containers werden gecreëerd. Deze zijn dus louter ter documentatie en moeten in principe niet meer uitgevoerd worden (zolang er gewerkt wordt met de bestaande containers). De MariaDB-containers moeten steeds eerst opgestart worden vooraleer de Moodle-container operatief kan zijn. Dit is het geval omdat Moodle enkel werkt wanneer een verbinding met de database actief is.

A.1.1.1 MariaDB-containers

Voor de database werd een MariaDB-*master-slave*-combinatie voorzien.

De MariaDB-*master*-container (`mariadb-master`):

```
docker run -d --name mariadb-master --net moodle-tier \
-e MARIADB_ROOT_PASSWORD=******** \
-e MARIADB_REPLICATION_MODE=master \
-e MARIADB_REPLICATION_USER=code_replication \
-e MARIADB_REPLICATION_PASSWORD=******** \
-e MARIADB_USER=code \
-e MARIADB_PASSWORD=******** \
-e MARIADB_DATABASE=mariadb-master \
-v /export/home1/code/persistence/mariadb_data:/bitnami \
bitnami/mariadb
```

[2] Meer uitleg over Docker-containers en -*images*: https://docs.docker.com/engine/docker-overview/#docker-objects.

De MariaDB-*slave*-container (`mariadb-slave`):

```
docker run -d --name mariadb-slave --link mariadb-master:master \
--net moodle-tier \
-e MARIADB_REPLICATION_MODE=slave \
-e MARIADB_REPLICATION_USER=code_replication \
-e MARIADB_REPLICATION_PASSWORD=******** \
-e MARIADB_MASTER_HOST=master \
-e MARIADB_MASTER_ROOT_PASSWORD=******** \
-v /export/home1/code/persistence/mariadb_slave_data:/bitnami \
bitnami/mariadb
```

A.1.1.2 Moodle-container

De Moodle-container (`moodle`):

```
docker run -d -p 80:80 -p 443:443 --name moodle --net moodle-tier \
-e MARIADB_PASSWORD=******** \
-e MARIADB_HOST=mariadb-master \
-v /export/home1/code/persistence/moodle_data:/bitnami \
bitnami/moodle
```

A.1.2 Status van containers opvragen

De status van de bestaande containers kan worden opgevraagd als volgt:

```
docker ps -a
```

A.1.3 Stoppen en starten van de containers

Het stoppen en (her)starten van een container kan aan de hand van de volgende opdrachten:

```
docker stop  <naam container>
docker start <naam container>
```

Het stoppen en herstarten is bijvoorbeeld nodig bij de Moodle-container wanneer het `SSL`-certificaat moet worden vernieuwd en Apache (dat maakt deel uit van de Moodle-container) daarom moet worden herstart (zie A.9.1).

Bij het herstarten van de Moodle-container is het belangrijk dat de MariaDB database al correct herstart is. Anders zal de Moodle-container niet correct opgestart geraken en moet het commando opnieuw manueel uitgevoerd worden.

A.1.4 Docker restart policies

Docker-containers worden niet automatisch herstart wanneer het systeem reboot. Een automatische herstart gebeurt uitsluitend als de juiste *restart policy*[3] voor de container wordt geselecteerd.

Voor Co-De werden de *restart policies* ingesteld als volgt:

```
docker update --restart=unless-stopped mariadb-master
docker update --restart=unless-stopped mariadb-slave
docker update --restart=unless-stopped moodle
```

De instelling `unless-stopped` zorgt ervoor dat de containers steeds automatisch herstarten, zelfs wanneer de machine gereboot werd. Enkel wanneer de containers manueel worden gestopt (zie A.1.3), zullen de containers niet automatisch herstarten.

A.1.5 Back-ups

Een back-up-script maakt dagelijks om 03:14:00 een back-up van de gepersisteerde data van de Moodle-container en MariaDB-container. Deze back-ups worden gedurende zeven dagen bewaard. De snapshots van de gegevens kunnen worden gebruikt wanneer de update van een plugin op Co-De niet succesvol is of een herstart faalt. Het systeem kan dan manueel naar een eerdere momentopname terug geplaatst worden. De inhoud van het back-up-script is weergegeven in het codevoorbeeld A.1.

[3] https://docs.docker.com/config/containers/start-containers-automatically/.

```
#!/bin/bash

# Moodle backup
# Get the current date
BACKUPTIME=‘date +%Y%m%d-%H.%M.%S‘

# Create a backup file using the current date in its name
DESTINATION=./backup/moodle_data.bkp.$BACKUPTIME

# The folder that contains the files that we want to
# backup
SOURCEFOLDER=/export/home1/code/persistence/moodle_data/

# Create the backup
rsync -av --exclude-from '/export/home1/code/exclude.txt'
    $SOURCEFOLDER $DESTINATION > log_bkp.txt

# Database backup
BACKUPTIME=‘date +%Y%m%d-%H.%M.%S‘

DESTINATION=./backup/mariadb_data.bkp.$BACKUPTIME
SOURCEFOLDER=/export/home1/code/persistence/mariadb_data

rsync -av $SOURCEFOLDER $DESTINATION >> log_bkp.txt

# Remove backups older than 7 days
find ~/backup/ -type d -ctime +7 \
-name 'moodle_data.bkp*' \
-execdir rm -r -- '{}' ; 2>&1 | grep -v find

find ~/backup/ -type d -ctime +7 \
-name 'mariadb_data.bkp*' \
-execdir rm -r -- '{}' ; 2>&1 | grep -v find
```

Codevoorbeeld A.1: De inhoud van backup_script.sh.

```
/moodle/moodledata/sessions
/moodle/moodledata/localcache
/moodle/moodledata/cache
/moodle/moodledata/temp
```

Codevoorbeeld A.2: De inhoud van exclude.txt. De inhoud van dit bestand is nodig tijdens de uitvoering van het back-up-script om na te gaan welke mappen niet moeten worden gekopieerd.

A.2 Onderhoudstaken van Moodle

Alle geplande taken van Moodle (zie figuur A.2) worden dankzij `cronjobs` op de server uitgevoerd, met het bestand `cron.php` dat zich in de map `admin` van Moodle bevindt. Dit bestand wordt niet automatisch uitgevoerd, maar moet nog zelf geconfigureerd worden door de sitebeheerder. Op Co-De wordt dit bestand elke tien minuten uitgevoerd, dankzij een `cronjob` in het script `moodle_script.sh` (zie codevoorbeeld A.3). `Cron.php` is beveiligd met een wachtwoord dat enkel gekend is door de site-beheerders. Zo wordt voorkomen dat een derde de geplande taken van Moodle kan laten uitvoeren.

```
1 #!/bin/bash
2
3 rm -f /export/home1/code/cron*
4
5 /usr/bin/wget -q  https://code.experiments.cs.kuleuven.be\
6 /admin/cron.php?password=********
```

Codevoorbeeld A.3: De inhoud van moodle_script.sh.

Name	Component	Edit	Last run	Next run	Minute	Hour	Day	Day of wee
Automated backups \core\task\automated_backup_task	Core	⚙	Friday, 18 May 2018, 1:50 PM Run now	Friday, 18 May 2018, 2:50 PM	50	*	*	*
Clean backup tables and logs \core\task\backup_cleanup_task	Core	⚙	Friday, 18 May 2018, 2:10 PM Run now	Friday, 18 May 2018, 3:10 PM	10	*	*	*
Remove expired cache entries \core\task\cache_cleanup_task	Core	⚙	Friday, 18 May 2018, 1:30 PM Run now	Friday, 18 May 2018, 2:30 PM	30	*	*	*
Background processing for caches \core\task\cache_cron_task	Core	⚙	Friday, 18 May 2018,	Friday, 18 May 2018,	50	*	*	*

Figuur A.2: Een schermafbeelding van de onderhoudstaken op de "site administration" van Co-De. Deze view kan gebruikt worden om de werking van het Moodle-script na te kijken.

A.3 Crontab

Met het commando `crontab` worden twee scripts (`cronjobs`) op vaste tijdstippen uitgevoerd. Enerzijds is er het back-up-script `backup_script.sh` (zie onderdeel A.1.5) dat een snapshot van de containers zal maken en opslaan (elke nacht om 03:14:00). Anderzijds is er het script dat de onderhoudstaken van Moodle verzorgt (om de 10'), namelijk `moodle_cron.sh` (zie A.2). Codevoorbeeld A.4 toont de instellingen van de twee `cronjobs`. Als er onverwachte output gegenereerd wordt door de `cronjobs` wordt die per e-mail verstuurd naar code.leerplatform@gmail.com.

Cronjobs kunnen gewijzigd worden door het volgende commando:

```
crontab -e
```

```
1 # Daily backup at 3:14 AM
2 14 3 * * * /bin/bash /export/home1/code/backup_script.sh \
3 2>&1 | mail -E -s 'Code backup cron output' \
4 code.leerplatform@gmail.com
5
6 # Moodle cronjob every 10'
7 */10 * * * * /bin/bash /export/home1/code/moodle_cron.sh \
8 2>&1 | mail -E -s 'Code moodle cron output' \
9 code.leerplatform@gmail.com
```

Codevoorbeeld A.4: De verschillende cronjobs van crontab.

A.4 Programmeeropdrachten: Jobe server

De programmeeropdrachten van CodeRunner vereisen dat er een eigen "Jobe" server gedeployed is om de ingestuurde code van leerlingen te compileren en testen. Jobe is eveneens beschikbaar via Docker dankzij de volgende *image*: https://hub.docker.com/r/trampgeek/jobeinabox/.

Het opstarten van de Jobe server gebeurt als volgt:

```
docker run -p 4000:80 trampgeek/jobeinabox:latest
```

Nadien moet de configuratie op Co-De nog aangepast worden door de sitebeheerder bij `Site administration` > `Plugins` > `CodeRunner` > `Jobe host`:

```
code.experiments.cs.kuleuven.be:4000
```

De container kan op dezelfde manier bediend worden als aangegeven in A.1.3. De *restart policy* is eveneens `unless-stopped` zoals bij de andere Docker-containers (zie A.1.4).

A.5 Animaties in Processing

Animaties zijn pagina-activiteiten waarop een interactieve animatie is ingesloten. Het kan gaan om een opdracht waarbij een bepaald probleem moet worden opgelost of om een interactieve illustratie van een bepaald concept. Deze activiteitensoort wordt uitgebreid toegelicht in 3.5.2.2 op pagina 45.

Ten eerste wordt in dit deel van de technische appendix dieper ingegaan op het ontwerp van de animaties zelf in "Processing". De werking van Processing wordt hier echter niet herhaald, maar kan men raadplegen via https://processing.org/. De bestaande animaties werden op een objectgerichte manier geprogrammeerd. De verschillende bouwstenen (knoppen, sliders, grafen...) uit deze animaties zijn dus ook bruikbaar voor nieuwe animaties. Ten tweede wordt kort herhaald hoe een afgewerkte animatie kan worden getoond op een webpagina en vervolgens in een pagina-activiteit van Moodle.

Deze uiteenzetting illustreert de werking van animaties aan de hand van een (vereenvoudigd) voorbeeld. Het resultaat van deze voorbeeld-animatie is hier terug te vinden: `$code/resources/paardenronde/opgave1/`.

A.5.1 Ontwerpen van animaties

A.5.1.1 main.pde

In elk Processing-project moeten de functies `setup()` en `draw()` gedefinieerd worden. Voor Co-De is het hoofdbestand steeds `main.pde` en worden deze functies daarin gedefinieerd. In de hoofding van `main.pde` worden alle objecten die (globaal) worden gebruikt in de animatie aangemaakt, zoals gebruikelijk in Java. Dit wordt geïllustreerd in codevoorbeeld A.5. Sommige van de objecten zijn van een ingebouwd type (`color`, `String`, `int`...) anderen zijn van eigen een gedefinieerde klasse (`Button`, `Raster`...).

De `setup()` functie bepaalt de grootte van het tekenblad en initialiseert objecten zoals wordt geïllustreerd in codevoorbeeld A.6. De functie `draw()` zorgt ervoor dat bepaalde objecten op het scherm verschijnen bij het starten van de animatie en bepaalt de kleur van de achtergrond en dergelijke. Codevoorbeeld A.7 toont hier een voorbeeld van.

In `main.pde` kunnen, indien nodig, bijkomende (hulp-)functies worden ondergebracht. Daarnaast is het ook mogelijk om functies zoals `mouseMoved()` of `mouseClicked()` te gebruiken. Deze functies worden automatisch uitgevoerd wanneer respectievelijk met de muis wordt bewogen of geklikt (binnen het venster van de animatie). Een voorbeeld van het gebruik van deze functies is terug te vinden in codevoorbeeld A.8. Men kan ook steeds gebruik maken van de positie van de muis die wordt gegeven door (`mouseX`, `mouseY`). De elementen die hier afhankelijk van zijn worden pas geüpdatet wanneer deze (onrechtstreeks) worden aangepast door `mouseMoved()`.

```
color red = #ff595e;
color green = #8ac926;
color blue = #1982c4;
color col; //Colors green/red when a solution is checked
String message1;
String message2;
int pogingen;

Raster raster;
Volgorde volgorde;

Button back;
Button reset;
Button test;
```

Codevoorbeeld A.5: Een voorbeeld van de hoofding van een main.pde bestand voor een animatie in Processing.

```
void setup() {
  size(950,570);
  f = createFont("AppleColorEmoji",13,true);

  raster = new Raster();
  volgorde = new Volgorde();
  back = new Button(25,520,190,"Ongedaan maken");
  reset = new Button(235,520,190,"Begin opnieuw");
  test = new Button(725,520,190,"Kijk na");

  message1 = "Klik op een vakje om het toe te voegen\naan de volgorde.
    ";
  message2 = "";
  pogingen = 0;
}
```

Codevoorbeeld A.6: Een voorbeeld van de inhoud van de setup() functie uit het main.pde bestand.

```
void draw() {
  background(255);

  volgorde.display();
  raster.display();
  back.display();
  reset.display();
  test.display();

  textAlign(CENTER);
  textFont(f,20);
  fill(0);
  text(message1 + "\n \n" + message2, 650, 140);
}
```

Codevoorbeeld A.7: Een voorbeeld van de inhoud van de draw() functie uit het main.pde bestand.

```
50 void mouseMoved() {
51   raster.update();
52   back.hover();
53   reset.hover();
54   test.hover();
55 }
56
57 void mouseClicked() {
58   raster.clicked();
59   if(back.clicked()){volgorde.undo(); col=white;}
60   if(reset.clicked()){volgorde.reset(); col=white;}
61   if(test.clicked()){volgorde.check(); pogingen++;}
62 }
```

Codevoorbeeld A.8: Een voorbeeld van het gebruik van de functies mouseMoved() en mouseClicked() in het main.pde bestand van een animatie in Processing.

A.5.1.2 Objecten

Naast het `main.pde` bestand bevatten de animaties op Co-De nog andere `.pde`-bestanden. Deze definiëren elk een objectklasse zoals bijvoorbeeld `Button` (zie codevoorbeeld A.9 en lijnen 12, 13 en 14 in codevoorbeeld A.5).

Ook deze bestanden starten met een hoofding waarin de klasse een naam krijgt en variabelen worden aangemaakt (lijnen 1 t.e.m. 4 in codevoorbeeld A.9). Daarna volgt een constructor (lijnen 6-12) die wellicht wordt opgeroepen vanuit de `setup()` functie (maar dit is niet verplicht).

Verder heeft een klasse meestal een `display()` functie die zegt wat er juist waar op het scherm moet worden getekend, geschreven... Voor de knop (`Button`) uit het voorbeeld is dat een blauwe rechthoek (met breedte `w` en hoogte `30`) en de gegeven tekst in het wit (`txt`). Beiden op de gegeven positie (`x`,`y`). De `hover()` en `clicked()` functie bepalen wat er moet gebeuren wanneer de muis over de knop beweegt (wordt grijs) of er op de knop wordt geklikt (`return true` zodat er een gevolg kan aan gekoppeld worden). Deze functies worden opgeroepen vanuit respectievelijk `mouseMoved()` en `mouseClicked()`. Zoals hierboven werd beschreven, maken deze functies gebruik van de coördinaten van de muisaanwijzer (`mouseX`, `mouseY`) om te bepalen waar de muis zich bevindt of waar er geklikt werd.

Aan de hand van verschillende *classes* werden een aantal generieke types ontwikkeld die worden gebruikt bij verschillende animaties. Deze types zijn wellicht ook handig om te gebruiken in nieuwe animaties. De verschillende zinvolle bouwstenen worden toegelicht in het volgende onderdeel en gelden als `API` voor nieuwe animaties. Elk van deze bouwstenen is opgebouwd zoals hierboven beschreven en men kan de implementatie ervan raadplegen via de bestaande animaties op Co-De. In een nieuwe animatie kan men best de `.pde`-bestanden van de nodige bouwstenen toevoegen aan het project. Op deze manier kan elke animatie onafhankelijk beheerd worden en kunnen (indien nodig) kleine wijzigingen worden gemaakt aan de bronbestanden.

```
class Button {
  float xpos; float ypos; float w; float h=30;
  String txt;
  color c;

  Button(float x, float y, float wi, String t){
    xpos = x;
    ypos = y;
    txt = t;
    w = wi;
    c = blue;
  }

  void display(){
    rectMode(CENTER);
    strokeWeight(2);
    fill(c);
    rect(xpos+(w/2),ypos+(h/2),w,h,7);
    textAlign(CENTER);
    textFont(f,20);
    fill(255);
    text(txt, xpos+(w/2), ypos+7+(h/2));
  }

  void hover(){
    if(mouseX>xpos && mouseX<xpos+w && mouseY>ypos && mouseY<ypos+h) {
      c = grey;
    }else{
      c = blue;
    }
  }

  boolean clicked(){
    if(mouseX>xpos && mouseX<xpos+w && mouseY>ypos && mouseY<ypos+h) {
   return true;}
    else{return false;}
  }
}
```

Codevoorbeeld A.9: Het volledige button.pde bestand zoals dat wordt gebruikt in de voorbeeld-animatie.

A.5.1.3 Beschikbare bouwstenen

Button

```
// Constructor
// (x,y)  the position
// w    the width, the height is alway 30
// t    the text to display
Button(float x, float y, float wi, String t){}

// Show the button
void display()

// Change the background when mouse hovered
void hover()

// Return true if clicked on the area of this button
boolean clicked()
```

Slider

```
// Constructor
// (px,py)  the position
// wi   the total width
Slider(float px, float py, float wi)

// Show a straight black line on the position with the given width
// Show a circle at the beginning of the line
// Show an empty String below the slider
void display()

// Update the position of the circle on the slider
void update(float nx, float ny)

// Get the absolute x-position of the circle
// Measured in according to the (0,0) origin
float getXabs()

// Get the relative x-position of the circle
// Measured in according to the origin of the slider (px)
float getXrel()

// Update the text below the slider
// Define your own text and changes of the text according to the
    position of the slider
void updateText()
```

Node

```
// Constructor
// (x,y)  the position
// ima_in the image (emoji) to show
// id_in  the unique identifier (not shown)
//      String for easy reference
Node(float x, float y, PImage ima_in, String id_in)

// Show circle of size 60 with image and black borders
void display()

// Change background color and enlarge image when hovered
void hover()

// Detect if the mouse has clicked on the surface
// Implement your own effect
void clicked()

// Get the unique identifier
String getid()

// Get the image
PImage getImage()
```

Edge

```
// Constructor
// Connect node1 and  node2 via an undirected edge
Edge(Node node1, Node node2)

// Show a straight black line between the center of the nodes
void display()
```

Vakje

```
// Constructor
// (x,y)  the position
// im   the image (emoji) to show
// sid    the unique identifier (not shown)
//      String for easy reference
Vakje(float x, float y, PImage im, String sid)

// Show square of size 80 with image and black borders
void display()

// Change backgroundcolor and enlarge image when hovered
void hover()

// Detect if the mouse has clicked on the surface
// Implement your own effect
void clicked()

// Get the unique identifier
String getid()

// Get the image
PImage getImage()
```

Een vakje kan ook in een bepaalde volgorde worden opgeslagen, bijvoorbeeld de volgorde waarin men erop klikt. Hiervoor werd een `volgorde.pde` bestand uitgewerkt, maar het gedrag hiervan is sterk afhankelijk van animatie tot animatie. Dit bestand integraal overnemen voor een nieuwe animatie is dus niet aan te raden. Het kan wel dienen als vertrekpunt voor gelijkaardig gebruik.

A.5.2 Embedden van animaties

Om lokaal ontwikkelde animaties online weer te geven wordt "Processing.js"[4] gebruikt. Via een `.html`-bestand kan de animatie nu online worden gebruikt. Figuur 3.13 (pagina 46) geeft de structuur van dit `.html`-bestand weer. Eens de animatie zichtbaar is op een webpagina kan ze via een `iframe` worden ingesloten in een pagina-activiteit (zie figuur 3.14 op pagina 46). Hierbij wordt ook steeds een link naar de oorspronkelijke animatie vermeld voor leerlingen bij wie de animatie niet zichtbaar is (zie "Indien de opdracht niet laadt..." in figuur 3.14).

Noot: De mogelijkheden in Processing en Processing.js zijn ongeveer gelijk, maar niet helemaal. Daarom is het aangeraden om nieuwe animaties eerst te testen in een tool die aangeeft welke operaties niet geldig zijn in Processing.js, bijvoorbeeld https://www.openprocessing.org/sketch/create. Het is ook mogelijk om de hele animatie in zo'n tool te ontwikkelen.

[4] http://processingjs.org/.

A.6 Plugin: relatieve voltooiing

"Relatieve voltooiing" of "relative completion" is een plugin die werd ontwikkeld in de context van Co-De, gebaseerd op de standaard voltooiing-plugin "availability completion". De volledige plugin kan gedownload worden via `$code/resources/relative_completion.zip`.

Figuur A.3 toont de hoog-niveau structuur van de plugin en codevoorbeelden A.10, A.11 en A.12 op de volgende pagina's tonen de belangrijkste plugin-specifieke broncode voor relatieve voltooiing.

- ▷ `relative_completion`
  - ▷ `classes`: de plugin-logica.
    - ▷ `condition.php`: de logica die bepaalt of een specifieke activiteit of sectie zichtbaar en/of toegankelijk zal zijn.
    - ▷ `frontend.php`: de logica die bepaalt welke opties de gebruiker (leerkracht, cursusbeheerder) kan selecteren voor een bepaalde activiteit of sectie.
  - ▷ `lang`: de verschillende taalpakketten die ondersteund worden door de plugin.
  - ▷ `tests`: de testen voor de plugin-logica.
  - ▷ `yui`: het formulier dat de beheerder van de les te zien krijgt bij het aanpassen van een activiteit of sectie. Dit beeldt de opties uit `frontend.php` af.
  - ▷ `version.php`: noodzakelijke meta-info over de plugin.

Figuur A.3: Het hoog-niveau overzicht van de structuur van de plugin "relatieve voltooiing". De hiërarchie is conform aan de opgelegde structuur van Moodle-plugins.

```
17 /**
18  * Version info.
19  *
20  * @package availability_relativecompletion
21  * @copyright 2017 Zimcke Van de Staey, Tobias Verlinde
22  * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
23  */
24
25 defined('MOODLE_INTERNAL') || die();
26
27 $plugin->version = 2018010500;
28 $plugin->requires = 2017110800;
29 $plugin->component = 'availability_relativecompletion';
```

Codevoorbeeld A.10: De inhoud van version.php uit de plugin relatieve voltooiing.

A.6.1 Condition.php

```
40 /** @var int IdType activity(0)/section(1) */
41 protected $idtype;
42
43 /** @var int Expected completion type (one of the COMPLETE_xx
       constants) */
44 protected $expectedcompletion;
45
46 /** @var array Array of modules used in these conditions for course */
47 protected static $modsusedincondition = array();
48
49 /**
50  * Constructor.
51  *
52  * @param \stdClass $structure Data structure from JSON decode
53  * @throws \coding_exception If invalid data structure.
54  */
55 public function __construct($structure) {
56   // Get idtype.
57   if (isset($structure->idtype) && is_number($structure->idtype)) {
58       $this->idtype = (int)$structure->idtype;
59   } else {
60       throw new \coding_exception('Missing or invalid ->idtype for
        relative completion condition');
61   }
62
63   // Get expected completion.
64   if (isset($structure->e) && is_number($structure->e)) {
65       if ($this->idtype == 1) {           // Case activity.
66           if (in_array($structure->e, array(0, 1, 2, 3))) {
67               $this->expectedcompletion = $structure->e;
68           } else {
69               throw new \coding_exception('Missing or invalid ->e
70               for relative completion condition');
71           }
72       } else if ($this->idtype == 0) {    // Case section.
73           if (in_array($structure->e, array(0, 1))) {
74               $this->expectedcompletion = $structure->e;
75           } else {
76               throw new \coding_exception('Missing or invalid ->e
77               for relative completion condition');
78           }
79       } else {
80           throw new \coding_exception('Invalid ->idtype for
81           relative completion condition');
```

Codevoorbeeld A.11: De initialisatie uit condition.php voor de plugin relatieve voltooiing.

```
86      /**
87       * Determines whether a particular item is currently available
88       * according to this availability condition.
89       *
90       * @param bool $not Set true if we are inverting the condition
91       * @param info $info Item we're checking
92       * @param bool $grabthelot Performance hint: if true, caches
     information
93       *   required for all course-modules, to make the front page and
     similar
94       *   pages work more quickly (works only for current user)
95       * @param int $userid User ID to check availability for
96       * @return bool True if this section or course module is available
97       *              False if this section or course module is
     unavailable or
98       *                    if there is currently no section or cm id
99       */
100     public function is_available($not, \core_availability\info $info,
     $grabthelot, $userid) {
101         $allow = false;
102
103         // Determine value of $allow.
104         if ($this->idtype == 1) {           // Case activities.
105             $allow = $this->is_available_activity($info, $grabthelot,
     $userid);
106         } else if ($this->idtype == 0) {     // Case sections.
107             $allow = $this->is_available_section($info, $grabthelot,
     $userid);
108         } else {
109             throw new \coding_exception('Missing or invalid ->idtype
     for completion condition');
110         }
111
112         // Reverse answer if $not is true.
113         if ($not) {
114             $allow = !$allow;
115         }
116
117         return $allow;
118     }
```

Codevoorbeeld A.12: De methode waarin wordt nagegaan of een bepaalde activiteit beschikbaar is voor de huidige gebruiker, in condition.php, uit de plugin relatieve voltooiing.

A.7 Plugin: voortgangsbalk

Bovenaan elke les wordt een voortgangsbalk getoond die weergeeft welke lesonderdelen al voltooid zijn. De voortgangsbalk maakt gebruik van de plugin "completion progress", maar de broncode werd licht gewijzigd. De plugin met de gewijzigde code is te vinden via `$code/resources/relative_completion.zip`.

De wijzigingen die werden aangebracht aan de voortgangsbalk worden hier meer in detail beschreven. Telkens als er over een blokje (activiteit) van de voortgangsbalk met de muis bewogen wordt, dan zal de titel van de bijhorende sectie onder de balk getoond worden. Hiervoor werd een extra methode geïmplementeerd in `lib.php` van de plugin, die de bijhorende sectie van een activiteit uit de database ophaalt. Die methode is weergegeven in codevoorbeeld A.13.

```
162 /**
163  * Returns the section title of the specified activity
164  *
165  * @retun string Name of section of the activity
166 */
167 function block_completion_progress_get_sectiontitle($activity) {
168   global $DB;
169
170   $sql = "SELECT cs.name FROM {course_sections} cs WHERE cs.id = :
      sectionnum";
171   $params = array('sectionnum' => $activity['sectionid']);
172
173   $section = $DB->get_record_sql($sql, $params);
174
175   return $section->name;
176 }
```

Codevoorbeeld A.13: De inhoud van lib.php uit de plugin completion progress.

Om de sectietitel vervolgens weer te geven op het scherm, werd de methode `block_completion_progress_bar($activities, ...)` aangevuld met de code uit codevoorbeeld A.14.

```
605   $content .= HTML_WRITER::empty_tag('br');
606   $content .= get_string('section','block_completion_progress').': ';
607   $content .= block_completion_progress_get_sectiontitle($activity);
608   $content .= HTML_WRITER::end_tag('div');
```

Codevoorbeeld A.14: De inhoud van lib.php uit de plugin completion progress.

Om de voortgangsbalk in het midden van de pagina, bovenaan de les te tonen, zijn er ook een aantal aanpassingen nodig in het thema (zie ook verder A.8). In `columns2.php` worden twee extra parameters `contentblockshtml` en `hascontentblocks` doorgegeven aan de `mustache template`-bestanden (zie regels 44-46 en 56 in A.15).

```
$bodyattributes = $OUTPUT->body_attributes($extraclasses);
$blockshtml = $OUTPUT->blocks('side-pre');
$contentblockshtml = $OUTPUT->blocks('content');// put blocks in the
    center region on course page
$hascontentblocks =
    strpos($contentblockshtml, 'data-block=') !== false;
$hasblocks =
    strpos($blockshtml, 'data-block=')!== false;

$regionmainsettingsmenu = $OUTPUT->region_main_settings_menu();
$templatecontext = [
    'sitename' => format_string($SITE->shortname, true, ['context' =>
    context_course::instance(SITEID), "escape" => false]),
    'output' => $OUTPUT,
    'contentblocks' => $contentblockshtml,
    'sidepreblocks' => $blockshtml,
    'hascontentblocks' => $hascontentblocks,
    'hasblocks' => $hasblocks,
    'bodyattributes' => $bodyattributes,
    'navdraweropen' => $navdraweropen,
    'regionmainsettingsmenu' => $regionmainsettingsmenu,
    'hasregionmainsettingsmenu' => !empty($regionmainsettingsmenu)
];
```

Codevoorbeeld A.15: De inhoud van columns2.php uit het thema code voor de plugin completion progress.

```
<section id="region-main" {{#hasblocks}}class="has-blocks"{{/hasblocks
    }}>
  {{#hascontentblocks}}
    {{{ contentblocks }}}
  {{/hascontentblocks}}
    <div class="card card-block">
        {{#hasregionmainsettingsmenu}}
            <div class="region_main_settings_menu_proxy"></div>
        {{/hasregionmainsettingsmenu}}
            {{{ output.course_content_header }}}
            {{{ output.main_content }}}
            {{{ output.activity_navigation }}}
            {{{ output.course_content_footer }}}
            </div>
</section>
{{#hasblocks}}
<section data-region="blocks-column" class="hidden-print">
    {{{ sidepreblocks }}}
</section>
{{/hasblocks}}
```

Codevoorbeeld A.16: De inhoud van column2.mustache uit het thema code voor de plugin completion progress.

A.8 Plugin: code (thema)

Het thema "code" is een *child-theme* van het standaard thema "Boost"[5]. In dit thema worden heel wat kleine wijzigingen aan het gedrag en de weergave van Moodle gebundeld. De recentste versie van deze plugin is beschikbaar via `$code/resources/theme_code.zip`.

De hoog-niveau structuur van de plugin wordt weergegeven in figuur A.4:

- ▷ `theme_code`
  - ▷ `config.php`: configuratie, inclusief mogelijkheid om elementen uit de configuratie van Boost te overschrijven.
  - ▷ `lib.php`: achtergrond callbacks
  - ▷ `settings.php`: instellingen, links naar andere files en functies.
  - ▷ `version.php`: informatie over de plugin (versie & afhankelijkheden).
  - ▷ `classes`
    - ▷ `core_renderer.php`: mogelijk om functies uit andere `renderers` te overschrijven.
  - ▷ `lang`: de verschillende taalpakketten die ondersteund worden door de plugin.
  - ▷ `layout`: pagina lay-outs als `.php`-bestanden.
  - ▷ `templates`: pagina lay-outs als `.mustache`-bestanden.
  - ▷ `pix`: afbeeldingen en favicon.
  - ▷ `scss`: style-bestanden.
    - ▷ `pre.scss`: stijlregels die worden toegevoegd vooraan.
    - ▷ `post.scss`: stijlregels die worden toegevoegd achteraan.

Figuur A.4: Het hoog-niveau overzicht van de structuur van de plugin "code". De hiërarchie is conform aan de opgelegde structuur van Moodle-plugins met thema's en de voorschriften voor het overschrijven van functies uit renderers.

Het gedrag van Moodle wordt soms bijgestuurd door één of meerdere functies in bepaalde `renderer`'s te overschrijven. De meeste activiteiten-soorten hebben zo een `renderer`. Het overschrijven gebeurt in de plugin door in `core_renderer.php` de desbetreffende `renderer` in te laden en de functie te herdefiniëren. Voor welke aanpassingen deze techniek gebruikt word, is beschreven in hoofdstuk 3 (p. 25). Een voorbeeld wordt weergegeven in codevoorbeeld A.17. Daar wordt de portaalpagina van de test-activiteit weggewerkt.

[5] https://docs.moodle.org/34/en/Boost_theme.

```
16 include_once($CFG->dirroot . "/mod/quiz/renderer.php");
17 class theme_code_mod_quiz_renderer extends mod_quiz_renderer {
18
19   // Line 779
20   // Skip the intro of Quizzes
21   public function view_page($course, $quiz, $cm, $context, $viewobj) {
22     redirect(
23       new moodle_url('/mod/quiz/startattempt.php',array('cmid' => $cm
     ->id, 'sesskey' => sesskey())),
24       '',
25       0
26     );
27     }
```

Codevoorbeeld A.17: Een voorbeeld van het overschrijven van een functie uit een andere renderer (mod_quiz_renderer).

Via de `.scss`-bestanden in de gelijknamige map worden heel wat `html`-elementen vormgegeven. Via de stijl-attributen worden verschillende elementen verborgen (`display:none;`), verkleind of vergroot. Ook het uniforme kleurgebruik wordt op deze manier bewerkstelligd. De vijf kleuren van Co-De worden toegekend aan variabelen die kunnen gebruikt worden doorheen de `.scss`-bestanden. Door het gebruik van die variabelen word het leerplatform vormgegeven.

De vijf kleurvariabelen:

```
$blue:   #1982c4;
$green:  #8ac926;
$yellow: #e8b835;
$red:    #ff595e;
$purple: #6a4c93;
```

Afdwingen van het kleurgebruik op Co-De (niet exhaustief):

```
$brand-primary:  $blue;
$brand-success:  $green;
$brand-info:     $purple;
$brand-warning:  $yellow;
$brand-danger:   $red;
$custom-control-checked-indicator-bg: $blue;
$custom-checkbox-indeterminate-bg:    $blue;
$state-success-bg:   $green;
$state-info-bg:      $blue;
$state-warning-bg:   $yellow;
$state-danger-bg:    $red;
$tag-info-bg:        $yellow;

.ishidden .tag-info{
   background-color: $red;
}
```

Via deze `.scss`-bestanden wordt ook de navigatie onderaan de lesonderdelen aangepast zoals wordt beschreven in 3.5.1.2 (p. 38):

```
a#prev-activity-link, a#next-activity-link{
   visibility: hidden;
   position: relative;
   white-space: nowrap;
   overflow: hidden;
}

a#prev-activity-link:after{
   visibility: visible;
   position: absolute;
   top: 0; left: 0;
   font-weight: bold;
   content: "< Vorige";
}

a#next-activity-link:after{
   visibility: visible;
   float: right;
   position: absolute;
   top: 0; right: 0;
   font-weight: bold;
   content: "Volgende >";
}
```

De aanpassingen in het thema die nodig zijn voor de voortgangsbalk worden in onderdeel A.7 besproken. Deze situeren zich voornamelijk in de bestanden in de mappen `layout` en `templates` van dit thema.

A.9 Beveiliging

A.9.1 SSL-certificaat

Co-De ondersteunt `HTTPS`, zodat een veilige uitwisseling van gegevens mogelijk is. Het `HTTPS`-verkeer op Co-De verloopt volgens poort `443` (standaard). Een gebruiker die de site via `HTTP` tracht te bereiken (poort `80`), wordt automatisch doorgestuurd naar de `HTTPS`-versie.

Het gesigneerde `SSL`-certificaat, dat nodig is om `HTTPS` mogelijk te maken, wordt periodisch aangeleverd door "Let's Encrypt" (https://letsencrypt.org/). Om gebruik te kunnen maken van Let's Encrypt, moet men kiezen voor een compatibele *client*. Gezien er geen rechten voorzien zijn om software te installeren op code@code.experiments.cs.kuleuven.be, werd opnieuw geopteerd voor een Let's Encrypt *client* die beschikbaar is als een Docker-*image*, namelijk "Zerossl". Deze *image* is beschikbaar op https://hub.docker.com/r/zerossl/client/.

Gezien de beperkte geldigheidsduur van een `SSL`-certificaat (twee maanden), is het belangrijk om tijdig een nieuw certificaat te genereren met de Zerossl-*image*. Een nieuw certificaat genereren kan aan de hand van de opdracht uit codevoorbeeld A.18:

```
1 docker run -it --name zerossl \
2 -v /export/home1/code/keys_and_certs:/data \
3 -v /export/home1/code/persistence/moodle_data/moodle/.well-known/acme-
      challenge:/webroot \
4 -u $(id -u) --rm zerossl/client \
5 --key account.key --csr domain.csr \
6 --csr-key domain.key --crt domain.crt \
7 --domains "code.experiments.cs.kuleuven.be" \
8 --generate-missing --path /webroot --unlink --live \
```

Codevoorbeeld A.18: De commando's voor het genereren van een nieuw SSL-certificaat door middel van Zerossl.

Het nieuwe certificaat is nu te vinden in `~/keys_and_certs/`.
Het certificaat en de sleutels kunnen vervolgens in de map `apache` geplaatst worden (zie codevoorbeeld A.19):

```
1 cp ~/keys_and_certs/domain.key ~/persistence/moodle_data/apache/conf/
      bitnami/certs/
2 cp ~/keys_and_certs/domain.key ~/persistence/moodle_data/apache/conf/
3
4 cp ~/keys_and_certs/domain.csr ~/persistence/moodle_data/apache/conf/
      bitnami/certs/
5 cp ~/keys_and_certs/domain.csr ~/persistence/moodle_data/apache/conf/
6
7 cp ~/keys_and_certs/domain.crt ~/persistence/moodle_data/apache/conf/
      bitnami/certs/
8 cp ~/keys_and_certs/domain.crt ~/persistence/moodle_data/apache/conf
```

Codevoorbeeld A.19: De commando's voor het kopiëren van de het nieuwe certificaat en de sleutel naar de Apache-map.

Bijlage B

Resultaten test 6 maart

B.1 Resultaten uit de vragenlijst buiten Co-De

Er werd aan de testgebruikers gevraagd om een vragenlijst in te vullen nadat ze de test hadden afgerond. Hierin werden een aantal vragen gesteld over hun ervaring.

	1	2	3	4	5
Platform					
Verliep het inloggen vlot?	2	5	4	2	4
Laadden de pagina's vlot?	0	0	1	5	11
Vond je de navigatie duidelijk binnen het platform?	0	1	1	5	10
Les					
Vond je de les leuk?	0	1	4	6	6
Vond je de les leerzaam?	0	1	5	9	2
Zou je de les aanraden aan anderen?	1	1	3	9	3

Tabel B.1: De vragen met een antwoord op een schaal van 1 t.e.m. 5 waarbij 1 staat voor "helemaal niet" en 5 voor "volledig wel". De cijfers in de tabel geven weer hoeveel testpersonen dit antwoord aangeduid hebben. Het totaal aantal antwoorden op elke vraag was 17.

	Ja	Nee
Heb je deze animatie kunnen zien? `$code/mod/page/view.php?id=4`	17	0
Kon je goed volgen waar je zat en of je klaar was met alle lesonderdelen?	11	6

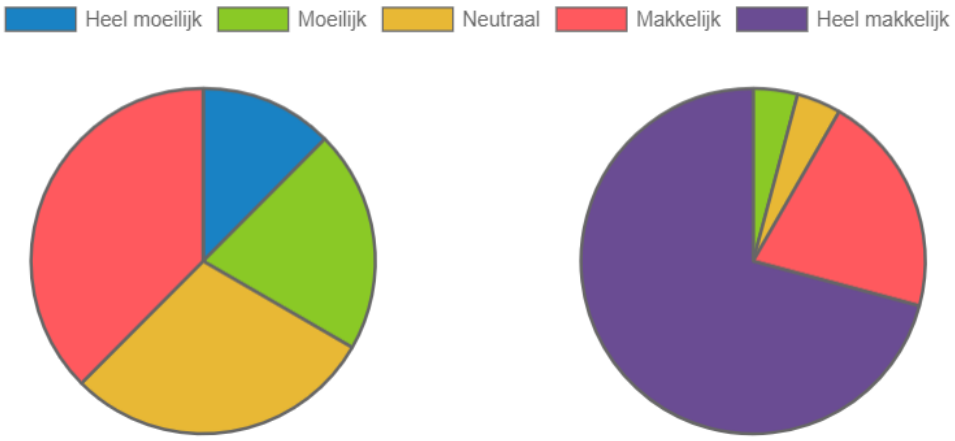
Tabel B.2: De ja/nee-vragen uit de vragenlijst. De cijfers in de tabel geven weer hoeveel testpersonen dit antwoord aangeduid hebben.

B.2 Antwoorden op de activiteiten in Co-De

Bij het volgen van de les moeten de gebruikers regelmatig antwoorden op vragen bij keuze-, opdracht- of test-activiteiten. Ook deze antwoorden zijn interessant om te bestuderen na een test en om verder mee aan de slag te gaan. De antwoorden op de belangrijkste vragen worden hieronder weergegeven.

B.2.1 Moeilijkheidsgraad opdrachten

De moeilijkheidsgraad van de twee hoofdopdrachten wordt bevraagd in twee keuze-activiteiten. De antwoorden worden hier gepresenteerd in de vorm van een taartdiagram zoals de leerlingen ze ook te zien krijgen.



(a) Verdeling van de antwoorden van 24 testpersonen op de vraag "hoe moeilijk vond je de paardenronde?".

(b) Verdeling van de antwoorden van 24 testpersonen op de vraag "hoe moeilijk vond je het stadsgidsprobleem?".

Figuur B.1: De antwoorden op de bevraging van de moeilijkheidsgraad van de twee problemen uit paardenronde & stadsgids.

B.2.2 Twee vakjes extra

Zestien testpersonen hebben de opdracht met de twee vakjes extra bereikt[1]. Om verder te kunnen moet men een antwoord invullen. Zes testpersonen gaven een leeg antwoord omdat ze in korte tijd de hele les wilden doorlopen en hier niet al te lang bij wilden blijven stilstaan. Van de tien overige personen vond 80% de juiste oplossing. Zij gingen, naar eigen zeggen, met pen en papier aan de slag. Ze breidden de graaf uit met twee extra knopen en zochten daarna een pad in die graaf. Dit blijkt dus een uitdagende en haalbare uitbreiding te zijn.

[1] `$code/mod/assign/view.php?id=19`.

B.2.3 Elementen computationeel denken

De antwoorden uit de vragenlijst over de elementen van computationeel denken[2] worden hieronder weergegeven. In het totaal vulden zeventien testgebruikers de vragenlijst in.

	De paardenronde	De stadsgids	Twee vakjes extra	Nergens
Waar heb je abstractie gebruikt?	7	5	6	4
Waar heb je veralgemeend?	2	1	13	2
Waar heb je decompositie gebruikt?	2	2	4	9
Waar heb je algoritmisch denken gebruikt?	7	5	12	2
Waar heb je geëvalueerd?	9	8	9	4

Tabel B.3: De antwoorden op de meerkeuzevragen uit de vragenlijst rond CD uit paardenronde & stadsgids. De getallen in de tabel zijn de hoeveelheid testpersonen die dat antwoord gaven. Één persoon kan meerdere antwoorden selecteren.

Hoe heb je aan abstractie gedaan?

- Maakt niet uit welke groente of fruit [*sic*] je neemt, stap moet gewoon kunnen stadsgids: maakt niet uit welke bezienswaardigheid je ziet.
- Het maakte niet uit wat welk vakje voorstelde.
- Door een schema te maken.
- Hertekenen geeft eenvoud.
- Op advies van jullie, heb ik een graaf uitgetekend. Ik heb een mooie professionele uitleg gegeven waarom er GEEN mogelijke oplossing was. De teleurstelling was dan ook groot dat jullie wel een oplossing aanboden... Voor een gemodificeerde graaf weliswaar. X heeft I en B als naburige knopen, niet I en H. Ditto redenering voor Y.
- De details over het paard of de haltes zijn niet belangrijk.
- De verschillende stukjes fruit heb je niet nodig. Je moet enkel weten dat je op elk vakje slecht [*sic*] een keer mag komen.
- De symbolen en het verhaaltje er rond [*sic*] kon je achterwegen laten bij het oplossen van een vraag.

[2] `$code/mod/feedback/view.php?id=21`.

Hoe heb je aan veralgemening gedaan?

- Door duidelijk te maken dat de twee soorten problemen eigenlijk hetzelfde zijn.
- Door dezelfde techniek te gebruiken als wanneer werd uitgelegd dat de twee opdrachten hetzelfde zijn.
- Graaf.
- Door de voorgestelde 'graaf'-techniek te gebruiken.
- Door de eerder opgedane ervaring kan je verder bouwen met dezelfde handelingen en denkpatronen uit te proberen.
- We hebben het paardenrondeprobleem veralgemeend tot het maken van een graaf en het oplossen van het stadsgidsprobleem dat deze graaf voorstelt.

Kan je nog andere problemen bedenken die je zou kunnen omvormen tot een route vinden in een graaf?

- Neen.
- Torens van Hanoi.
- Het is laat op de dag..misschien het verloop van de dag...hoe kan ik mijn werk gedaan krijgen door in alle lokalen éénmaal geweest te zin en daarbij de minste weg [*sic*] af te leggen.
- Nog vakjes bijplaatsen.
- Pizza-delivery.
- Een verkoper die allerlei klanten moet bezoeken.

Hoe heb je decompositie gebruikt?

- Eerst een schema maken, dan proberen dat de lijnen elkaar niet raken en dan een route zoeken.
- Door herschikking (is toch ook decompositie) kreeg ik inzichten in de probleemstelling.
- Deelproblemen opgelost door eerst de knopen die eerst of laatst moeten bezocht worden eerst te doorlopen. Dit vereenvoudigt de graaf tot een kleinere graaf.
- Het paardenronden probleem [*sic*] werd opgedeeld in het maken van een graaf en het oplossen van het stadsgidsprobleem dat hieruit voortkwam.

Hoe heb je algoritmisch denken gebruikt?

- Eerst een schema maken, dan de lijnen niet doen kruisen en als laatste een route zoeken.
- Nadenken over wat de volgende stap, de tweede-volgende stap enz kan zijn....in je hoofd een aantal stappen maken die je kan onthouden....ze dan doen en dan verder kijken welke stappen je daarna kan maken tot op een niveau dat je ze nog kan onthouden in je hoofd....papier had ik niet ter beschikking om alles te visualiseren.
- Omdat ik bij de twee vakjes extra niet langer intuïtief naar een oplossing zocht maar de opgedane inzichten heb gebruikt om tot een oplossing te komen.
- Stappen in het omvormen van de figuur tot een graaf.
- Je probeert niet in het wild, wel beredeneerd. D.w.z. dat je probeert een zekere logica te brengen in je denken en doen. Dit houdt vaak een repetitief handelen in volgens een bepaald patroon, lees algoritme.
- Het maken van de graaf gebeurt in een stappenplan.
- Je probeert een stappenplan uit te dokteren om tot het goede resultaat te komen.

Hoe heb je geëvalueerd?

- Bij de paardenronde besefte ik, na wat testen, dat gewoon iets proberen niet ging lukken. Bij de twee vakjes extra evalueerde ik (met wat hulp) at [*sic*] ik gebruik kon maken van de reeds aangereikte tekening. Dit was efficiënter dan zelf een tekening proberen te maken waarbij de lijnen niet kruisen.
- Door na te kijken of ik het juiste aantal stations had aangedaan.
- Nakijken dat je antwoord aan de vraag voldoet, door te zien dat alles aan bod gekomen is en te tellen.
- Vanaf dat je bij de paardenronde ergens was, wist je dat dit genoeg was om een oplossing te hebben. Je hoefde de laatste stappen niet meer expliciet te doorlopen. Bij de stadsgids kon je ook zien dat eenmaal je de binnenring bent doorlopen, je de buitenring vanzelf afgaat. Bij twee vakjes extra kwam er evaluatie bij het maken van de graaf van pas en bij het oplossen van de stadsgidskaart die hieruit voortkwam.
- Vanaf dat je de eerste stappen juist hebt komt de laatste sowieso uit.
- Door een stap te zetten moet je lijken [*sic*] of het nog mogelijk is om tot het goede resultaat te komen.

Bijlage C

Mock-up van de marslander

Deze appendix bevat de laatste versie van de mock-up voor de marslander. Deze is gemaakt in Microsoft PowerPoint en vormt de basis voor de verdere uitbouw van de les op Co-De. Zoals wordt beschreven in 4.2.2.1 (p. 73) is deze mock-up slechts een ontwerp en wijkt de eigenlijke les op Co-De hier geregeld van af.

De slides uit de PowerPointpresentatie worden per drie op een pagina weergegeven. De eerste twee dia's bevatten een ontwerp van de basisinformatie voor leerkrachten die later bovenaan in de les te zien zal zijn. Daarna volgt de les zelf in chronologische volgorde volgens het standaard leerpad.

Het eindresultaat van deze les is terug te vinden via `$code/course/view.php?id=9`. De les wordt uitgebreid besproken in 4.5 (vanaf pagina 91).

Over deze les

In deze les krijgen de leerlingen een opgave om op een gegeven marslandschap zoveel mogelijk bodemmonsters te verzamelen met een marsrover die enkele regels moet volgen. Eerst wordt dit met de hand opgelost voor enkele voorbeeld-marslandschappen en daarna wordt ingezoomd op enkele algoritmes met elk hun voor- en nadelen.

Dynamisch programmeren is een goed voorbeeld van decompositie:

Het probleem decomposeren in kleinere problemen van dezelfde soort.
Tot je komt bij een basisgeval.
Dan de oplossingen van de deelproblemen gebruiken om tot een volledige oplossing te komen.

CD vaardigheden:
Decompositie
Algoritmisch denken
Evalueren

Concepten:
Exhaustief zoeken
(Gulzig zoeken)
Dynamisch programmeren

Mogelijke leerlijnen

Keuzes:

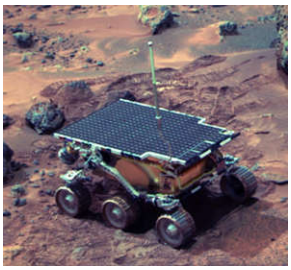
- 1, 2 of 3 algoritmes
- Volgorde algoritmes
- Ingaan op algemene formule aantal reiswegen of niet
- Programmeren al dan niet

De marslander

Op 4 juli 1997 landde de Mars Padvinder (Mars Pathfinder) van NASA op Mars. Het ruimtetuig droeg de eerste, vanop afstand bestuurbare rover met zich mee, genaamd Sojourner. De opdracht van Sojourner was om het marslandschap te verkennen en om beeldmateriaal en bodemstalen te verzamelen.

Video?
https://www.youtube.com/watch?v=o6As1LZzGzY

In deze les proberen we een goede weg te vinden die de rover kan afleggen op het marslandschap, om zoveel mogelijk informatie te verzamelen.



https://en.wikipedia.org/wiki/Sojourner_(rover)

https://en.wikipedia.org/wiki/Mars_landing

Opgave

Op de figuur aan de rechterkant zie je een voorstelling van een stukje marslandschap, opgedeeld in vierkante delen van 1 op 1 meter. We weten op voorhand hoeveel monsters er op elk stukje liggen en zullen nu trachten zoveel mogelijk monsters te verzamelen.

Het doel is dus om een reisweg voor de rover uit te stippelen en onderweg zoveel mogelijk monsters te verzamelen.

Je weet het volgende over de rover:
- De rover start in de **noordwestelijke hoek**
- De rover eindigt in de **zuidoostelijke hoek**
- De rover kan zich enkel in de richtingen oost en zuid bewegen

1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

Welke weg kan je volgen om zoveel mogelijk monsters te verzamelen? Hoe pak je dit aan?

Je antwoord…

Exhaustief zoeken

Exhaustief zoeken betekent dat we alle mogelijkheden proberen af te gaan en steeds de beste oplossing bijhouden.

In dit geval proberen we dus alle paden van de **noordwestelijke hoek** naar de **zuidoostelijke hoek** te vinden die voldoen aan de regels en steeds het aantal monsters uit te tellen. Het pad met de meeste monsters is dan de beste route voor de rover.

1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

Hoeveel mogelijke routes denk je dat er zijn voor dit stukje van 4m op 4m?

Je antwoord… **(20)**

Hoeveel denk je dat dat er zouden zijn voor 5m op 5m?

Je antwoord… **(70)**

Exhaustief zoeken (leerkracht)

Een woordje uitleg voor de leerkracht over het aantal routes.
De leerkracht kan dan beslissen om hier al dan niet dieper op in te gaan.

1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8
1	3	6	10	15	21	28	36
1	4	10	20	35	56	84	120
1	5	15	35	70	126	210	330
1	6	21	56	126	252	462	792
1	7	28	84	210	462	924	1716
1	8	36	120	330	792	1716	3432

$$\binom{n}{r} = \frac{n!}{r! \cdot (n-r)!}$$

$$\text{aantalroutes}(n) = \binom{2(n-1)}{(n-1)} = \frac{(2(n-1))!}{(n-1)! \cdot (n-1)!}$$

$$\text{aantalroutes}(4) = \binom{6}{3} = \frac{6!}{3! \cdot 3!} = \cdots = \frac{6 \cdot 5 \cdot 4}{3 \cdot 2 \cdot 1} = \frac{120}{6} = 20$$

Gulzig zoeken

Pas later bijgekomen, rechtstreeks op Co-De zodat hier samen aan kan gewerkt worden.

Slimmer aanpakken

Bij het exhaustief zoeken stoten we op twee problemen:

1. Het aantal mogelijke routes stijgt snel wanneer het gebied vergroot.
2. We moeten vaak dezelfde vakjes bekijken en de waarde optellen om het aantal monsters te berekenen. Elke keer als een vakje voorkomt in een route moeten we dezelfde bewerking herhalen, namelijk de waarde optellen bij het aantal monsters.

1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

Bespreek met je klasgenoten of je andere manieren kan bedenken om dit probleem op te lossen.
Probeer daarbij om te focussen op de bovenstaande twee problemen en deze te verminderen of weg te werken.

Je antwoord…

Een vergelijking

Dat het aantal mogelijke routes snel stijgt wanneer het gebied vergroot, daar kunnen we niets aan veranderen want dat is een vaststaand feit.

Wat je wel kan doen is proberen om een methode te vinden die niet al de routes moet afgaan om de beste oplossing te vinden. Indien je zo een methode kan vinden, heb je ook meteen het tweede probleem opgelost, aangezien je niet zoveel keer naar een vakje kijkt als er routes passeren.

1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

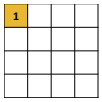
Bij onze nieuwe oplossingsmethode gaan we proberen om niet alle routes te overlopen maar ons te focussen op de goede/beste routes.

We zullen hiervoor **elk vakje** één keer onder de loep nemen in plaats van elke route. Het aantal vakjes stijgt ook naarmate het gebied groeit, maar veel minder fel dan het aantal mogelijke routes.

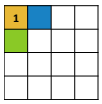
We zijn dus op zoek naar een **volgorde** waarmee we de vakjes kunnen doorlopen en een **vergelijking** die zegt hoeveel monsters we maximaal kunnen vinden op weg naar dat vakje.

Een vergelijking

Het staat vast dat we moeten beginnen in **noordwestelijke hoek**. We verzamelen dus sowieso één monster op die plek.



Vul nu op de **vakjes die je kan bereiken door één stap te zetten** (oost of zuid) het maximaal aantal monsters in dat je op weg daarnaartoe kan verzamelen.



Stel ook een vergelijking op waarmee je deze getallen kan berekenen aan de hand van de informatie die we al hebben:

- De **opgave** met het aantal monsters per vakje (w)
- In de **noordwestelijke hoek** verzamelen we maximum 1 monster

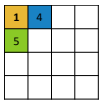
1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

Opgave

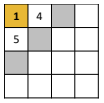
Een vergelijking

Het maximaal aantal monsters dat je kan verzamelen na één stap is 4 indien je naar het oosten gaat en 5 indien je naar het zuiden gaat.

Je kan dit berekenen door de som te maken van waar je vandaan kwam en wat er ligt op het nieuwe vakje.



Vul nu opnieuw op de **vakjes die je kan bereiken door één stap te zetten** (oost of zuid) het maximaal aantal monsters in dat je op weg daarnaartoe kan verzamelen.



Gebruik waar mogelijk je vergelijking uit de vorige opdracht of stel deze bij indien nodig. Je kan hierbij gebruik maken van alle informatie die we al hebben:

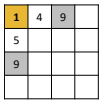
- De opgave met het aantal monsters per vakje
- Het maximum aantal monsters na 0 en 1 stappen

1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

opgave

Een vergelijking

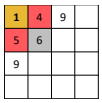
De vakjes aan de rand van het gebied kan je vinden met dezelfde vergelijking als daarnet. Namelijk door het optellen van het maximum aantal monsters van de plek vanwaar je komt en het aantal monsters op het vakje zelf.



1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

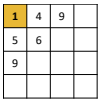
Opgave

Bij het andere vakje kunnen we via 2 routes geraken en op elk van deze routes nemen we misschien een ander aantal monsters:
- eerst oost, dan zuid -> 5 monsters
- eerst zuid, dan oost -> 6 monsters



Om te weten hoeveel monsters we maximaal kunnen vinden op weg naar en op dit vakje moeten we kijken naar de vakjes in het **noorden** en in het **westen** en bij het maximum van die twee het aantal monsters in dit vakje optellen.

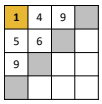
In dit geval is de route die van het westen komt beter dan die die van het noorden komt. We nemen dus de route via het westen en tellen bij het maximum aantal van daar (5) het aantal monsters op het vakje zelf op (1).

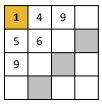


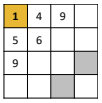
Een vergelijking

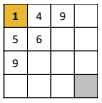
Ga zo steeds 1 stap verder en kijk naar het noorden en het westen om te bepalen wat het maximum aantal monsters is tot daar toe.

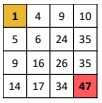
Kies daarvan het hoogste en tel er het aantal monsters op het vakje zelf bij op.











1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

opgave

Wat is het maximum aantal monsters dat je kan verzamelen als je toekomt bij het einde in de **zuidoostelijke hoek**?

Je antwoord...

Wat is de route die je daarvoor moet volgen?

Formuleer je antwoord als een opeenvolging van de stappen.
Bv: Oost, Zuid, Oost, Zuid, Oost, Zuid

De maxmonstertabel

De tabel die we hebben opgesteld in de vorige oefeningen noemen we de **maxmonstertabel**. Ze bevat namelijk voor elk gebiedje het maximum aantal monsters dat je kan verzamelen op weg naar dat stukje.

Het antwoord van de hele oefening kan je dan aflezen in de maxmonstertabel in de **zuidoostelijke hoek**.

We kunnen dus het volledige probleem oplossen door een vergelijking op te stellen voor de maxmonstertabel:

1	4	9	10
5	6	24	35
9	16	26	35
14	17	34	**47**

Maxmonstertabel
mam

1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

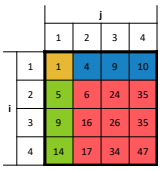
Opgave
w

1	4	9	10
5	6	24	35
9	16	26	35
14	17	34	**47**

Maxmonstertabel

De maxmonstertabel

We kunnen dus het volledige probleem oplossen door een vergelijking op te stellen voor de maxmonstertabel (mam):



1	4	9	10
5	6	24	35
9	16	26	35
14	17	34	**47**

Maxmonstertabel (mam)

1	3	5	1
4	1	15	11
4	7	2	0
5	1	8	**12**

Opgave (w)

Voor de noordwestelijke hoek:
mam(1,1) = w(1,1)

Voor de randen:
mam(1,j) = w(1,j) + mam(1,j-1)
mam(i,1) = w(i,1) + mam(i-1,1)

Voor alle andere vakjes:
mam(i,j) = w(i,j) + max[mam(i-1,j) , mam(i,j-1)]

De maxmonstertabel

Stel nu de maxmonstertabel op voor deze maanlandschappen.

Let goed op de verschillende stappen die je volgt, die kan je gebruiken bij het programmeren van de oplossing in de volgende opdracht.

1	4	3
4	2	7
7	3	**8**

2	10	2	6
5	0	6	0
4	3	2	1
6	1	7	**1**

2	3	5	2	6
5	0	1	6	0
0	7	10	2	4
4	3	5	2	1
6	1	8	7	**3**

1	5	6	8	2	0	3
2	6	9	4	5	8	2
1	0	0	3	5	6	8
2	1	1	2	5	6	7
4	4	6	9	5	8	7
4	0	1	2	3	4	2
2	3	9	6	1	5	**2**

Programmeren

Nu gaan we een python programma schrijven dat de maxmonstertabel berekent aan de hand van de vergelijkingen die we hebben opgesteld.

Als input krijg je:
- N: de grootte van het gebied in meter
- De opgave (w): een vierkante matrix van N op N met hoeveel monsters er waar liggen

De gevraagde output is:
- maxmon: de maxmonstertabel, een vierkante matrix van N op N
- resultaat: het antwoord op de vraag “hoeveel monsters kan je maximaal verzamelen?”

1. Test je implementatie op de gegeven marslandschappen.
2. Bedenk ook zelf eens een marslandschap waarop je je code kan testen.

Noot: 2 (of meer) versies aanbieden. Een volledige blanco, één met een hint naar de lesstructuur, …

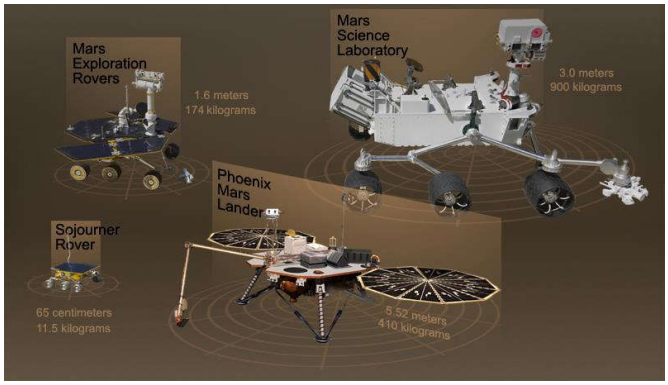
Uitbreidingen

Wat als…

- … er een negatief aantal monsters zou zijn?
  Er ligt op dat vakje bijvoorbeeld een chemische stof die gevonden monsters kapotmaakt.
- … je op sommige vakjes niet kan of mag komen?
  Het landschap is er bijvoorbeeld te ruw om over te kunnen rijden met de rover.
- … je twee rovers zou hebben die op dezelfde plaats beginnen en eindigen?
- … je rover ook naar het westen mag rijden?

Einde

Wist je dat de Sojourner slechts het aller begin was van de marsrovers?



Bibliografie

[1] B. Bastiaensen en J. De Creamer. Zo denkt een computer. https://onderwijs.vlaanderen.be/sites/default/files/atoms/files/Zo-denkt-een-computer.pdf, 2017. Uitgave in het kader van het Codefestival 2017.

[2] Bebras. Bebras Computing Challenge 2017. http://www.bebraschallenge.org/. Geraadpleegd: 02/11/2017.

[3] S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari, en K. Engelhardt. Developing Computational Thinking in Compulsory Education. JRC Science for Policy Report, European Commission, 2016.

[4] P. Bradford, M. Porciello, N. Balkon, en D. Backus. The Blackboard Learning System: The Be All and End All in Educational Instruction? *Journal of Educational Technology Systems*, 35(3):301–314, 2007.

[5] Centrum voor Blended Learning van Kulak. Achtergrondinformatie bij het 4C/ID-model. https://www.kuleuven-kulak.be/BlendedLearning/Blended%20learning/achtergrondinformatie-bij-het-4c-id-model. Geraadpleegd: 12/03/2018.

[6] Centrum voor Blended Learning van Kulak. Het vier componenten instructiemodel (4C/ID-model). https://www.kuleuven-kulak.be/BlendedLearning/Blended%20learning/4c-id-model. Geraadpleegd: 12/03/2018.

[7] Computing At School. Computing in the national curriculum. A guide for secondary teachers, 2014.

[8] ConXioN. Wat is GDPR? https://gdpr-eu.be/wat-is-gdpr/, 2017. Geraadpleegd: 20/04/2018.

[9] A. Csizmadia, P. Curzon, M. Dorling, S. Humphreys, T. Ng, C. Selby, en J. Woollard. Computational thinking, A guide for teachers. http://community.computingatschool.org.uk/files/8550/original.pdf, 2015.

[10] CSTA. Computer Science Teachers Association. https://www.csteachers.org/page/About, 2017. Geraadpleegd: 07/05/2018.

[11] R. Culatta. Instructional Design. https://www.instructionaldesign.org, 2018. Geraadpleegd: 17/05/2018.

[12] P. Curzon en P. W. McOwan. *The power of computational thinking: games, magic and puzzles to help you become a computational thinker*. World Scientific Publishing Europe Ltd., Londen, 2017.

[13] V. Dagiene en S. Sentance. It's Computational Thinking! Bebras Tasks in the Curriculum. In *ISSEP 2016*, ISSEP, pages 28–39, New York, NY, USA, 2016. Springer International Publishing AG.

[14] P. J. Denning. Remaining Trouble Spots with Computational Thinking. *Communications of the ACM*, 60(6):33–39, Mei 2017.

[15] G. Giselle Vercauteren. Onderwijssysteem niet afgestemd op mogelijkheden en noden van hedendaagse digitalisering. http://www.knack.be/s/r/c/566409, 2015. Geraadpleegd: 14/05/2018.

[16] A. Hoskey en S. Zhang. Computational Thinking: What Does It Really Mean for the K-16 Computer Science Education Community. *Journal of Computing Sciences in Colleges*, 32(3):129–135, Januari 2017.

[17] S. P. Jones. Computing at School: the state of the nation. A report of the Computing at School Working Group For the UK Computing Research Committee, Computing at School, 2009.

[18] Koninklijke Nederlandse Akademie van Wetenschappen. Digitale geletterdheid in het voortgezet onderwijs. Vaardigheden en attitudes voor de 21ste eeuw. Advies van de KNAW-Commissie Informatica in het voortgezet onderwijs, Koninklijke Nederlanse Akademie van Wetenschappen (KNAW), December 2012.

[19] R. E. Mayer, redacteur. *The Cambridge Handbook of Multimedia Learning*. Cambridge Handbooks in Psychology. Cambridge University Press, Cambridge, 2de editie, 2014.

[20] R. E. Mayer en R. Moreno. A cognitive theory of multimedia learning: Implications for design principles. *Journal of Educational Psychology*, 91(2):358–368, 1998.

[21] Moodle. Moodle Statistics. http://moodle.net/stats/, 2017. Geraadpleegd: 24/12/2017.

[22] Moodle. Histoy of Moodle. https://docs.moodle.org/34/en/History, 2018. Geraadpleegd: 12/04/2018.

[23] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA, 1980.

[24] G. Pólya. *How to Solve It*. Princeton University Press, 1945.

[25] G. Samaey en J. Van Remortel. Informaticawetenschappen in het leerplichtonderwijs. Standpunten van de Klasse van Technische Wetenschappen en de Jonge Academie, Koninklijke Vlaamse Academie van België voor Wetenschappen en Kunsten, in samenwerking met de Jonge Academie, November 2014.

[26] F. K. Sarfo en J. Elen. Developing technical expertise in secondary technical schools: The effect of 4C/ID learning environments. *Learning Environments Research*, 10:207–221, 2007.

[27] Schwitch. Wat is GDPR? http://scwitch.be/toolkit/, 2018. Geraadpleegd: 05/04/2018.

[28] Smartschool. Hét digitaal schoolplatform. http://www.smartschool.be/. Geraadpleegd: 12/04/2018.

[29] The Royal Society. Shut down or restart? The way forward for computing in UK schools. The Royal Academy of Engineering, 2012.

[30] N. Tollervey. Python in Education. http://www.oreilly.com/programming/free/python-in-education.csp, 2018. Geraadpleegd: 12/04/2018.

[31] J. van der Hoeven en B. Hoogland. Wat is het effect van het gebruik van het 4C/ID model op de lesuitvoering van docenten en op de kwaliteit van de les? https://www.nro.nl/kennisrotondevragenopeenrij/effect-4cid-model-op-lesuitvoering-docenten-en-kwaliteit-lessen/. Geraadpleegd: 28/04/2018.

[32] J. J. G. Van Merriënboer. 4C/ID als onderzoekskader. Slides uit de presentatie voor een lezing op de ALTUS-dag op 22 november 2011 in Leuven. https://associatie.kuleuven.be/altus/seminaries/1112/221111/4C-ID.pdf, 2011.

[33] J. J. G. Van Merriënboer en P. A. Kirschner. *Ten steps to complex learning: a systematic approach to four-component instructional design*. Routledge, New York, NY, 2de editie, 2013.

[34] J. Visser. Deze computerwetenschapper doet een voorzet voor écht programmeeronderwijs. https://decorrespondent.nl/7401/deze-computerwetenschapper-doet-een-voorzet-voor-echt-programmeeronderwijs/588031653-1dc9ffff, 2017. Geraadpleegd: 24/11/2017.

[35] J. M. Wing. Computational Thinking. *Communications of the ACM*, 49(3):33–35, Maart 2006.

[36] WiPSCE. WiPSCE - 13th Workshop in Primary and Secondary Computing Education. http://www.informatikdidaktik.de/wipsce2018, 2018. Geraadpleegd: 02/05/2018.

Fiche masterproef

Studenten: Zimcke Van de Staey
Tobias Verlinde

Titel: **Co-De**: een digitaal leerplatform voor computationeel denken

Engelse titel: **Co-De**: A Digital Learning Platform for Computational Thinking

UDC: 681.3

Korte inhoud:
This project researches the development and deployment of an online learning platform (Co-De) for computational thinking (CT). The envisioned group of end users are teachers with a scientific background and their students from secondary schools. Different authors argue that computational thinking is equally important in today's world as essential skills such as reading or writing. They also insist on making CT a part of the compulsory curriculum. In Flanders (Belgium) CT is not yet a mandatory component of secondary compulsory education. Co-De is based on Moodle, an extensive open source learning management system. Within the scope of this thesis, a collection of four courses (case studies) were designed for the platform. Every course discusses a specific problem or puzzle and introduces related computer science concepts (e.g. directed and undirected graphs, dynamic programming, search heuristics and algorithms, etc.). Each course (taking 1-4 study periods of 50') focuses on two or three aspects of computational thinking. The aspects of CT that are taken into consideration are the following: abstraction, generalization, decomposition, algorithmic thinking and evaluation. This view of CT is based on previous findings from Computing At School. The platform makes use of different types of content to support multimedia learning: text and illustrations, video examples, animations with user interaction, evaluation forms and quizzes for exercises, etc. An exercise can be in different formats such as so-called "unplugged" activities, online questions and programming exercises within the platform. A main aspect of Co-De is the flexibility with which the teacher can adjust the course content. Different learning paths are suggested for all courses, from which the teacher can select one or define his own. To assemble and evaluate the learning paths, the 4C/ID-model was used as a general framework. Some informal usability tests were conducted to optimize the methods of navigation on the platform. In the back end, Co-De is deployed with Docker, which allows it to run platform-independently.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdoptie artificiële intelligentie

Promotoren: Prof. dr. Bart Demoen
Prof. dr. Bern Martens

Assessoren: Prof. dr. ir. Yolande Berbers
Prof. dr. ir. Hendrik Blockeel