Vaardig worden in computational thinking met WeDo 2.0

In dit hoofdstuk ontdek je hoe je WeDo 2.0 kunt gebruiken om computational thinking-vaardigheden te ontwikkelen in een wetenschappelijke context.

Vaardig worden in computational thinking met LEGO® Education WeDo 2.0-projecten

LEGO® Education presenteert je met trots deze projecten, die speciaal zijn ontwikkeld om in de basisschool te gebruiken om de computational thinking-vaardigheden van de leerlingen te ontwikkelen.

Computational thinking heeft betrekking op een set vaardigheden die iedereen kan toepassen om alledaagse problemen op te lossen. In WeDo 2.0 worden deze vaardigheden tijdens de verschillende fasen van elk project ontwikkeld. In elk project zijn mogelijkheden voor ontwikkeling geïdentificeerd. Jij bepaalt zelf welke het meest relevant zijn voor jou en je leerlingen.

In elk project van WeDo 2.0 wordt het gebruik van LEGO stenen gecombineerd met een programmeertaal met pictogrammen, waardoor je leerlingen problemen leren op te lossen en kennismaken met programmeerprincipes.

Met WeDo 2.0 worden de computational thinking-vaardigheden van je leerlingen ontwikkeld via codeeractiviteiten waarmee hun creaties tot leven worden gebracht, die zorgen voor veel leerplezier en zin om meer te ontdekken.

Informatica, computational thinking, programmeren

Wetenschap en techniek bestaan al sinds het begin van de mensheid, maar informatica is een veel jongere discipline. Deze jonge discipline heeft niet alleen een invloed gehad op de manier waarop we wetenschap en techniek benaderen, maar ook op onze levensstijl.

Informatica is een STEM-discipline en heeft veel van dezelfde eigenschappen als wetenschap, technologie, techniek en wiskunde.

Alle STEM-disciplines bieden de mogelijkheid om een bepaalde denkwijze en een set levenslang te gebruiken competenties te ontwikkelen. Onder deze competenties vinden we het vermogen om vragen te stellen, oplossingen te bedenken en resultaten te presenteren.

Eén van deze competenties is computational thinking. Het is een denkwijze en een manier om problemen op te lossen.

Computational thinking is te omschrijven als een groep vaardigheden, waarvan één algoritmisch denken is. Coderen betekent het creëren van een algoritme.

Coderen is daarom één van de manieren om computational thinking te ontwikkelen in een STEM-context.

STEM-disciplines

Wetenschap, technologie, techniek, wiskunde en informatica

Een bepaalde denkwijze en een set levenslang te gebruiken competenties ontwikkelen

1. Vragen stellen en problemen oplossen.
2. Modellen gebruiken.
3. Prototypes ontwerpen.
4. Onderzoeken.
5. Gegevens analyseren en interpreteren.
6. Computational thinking gebruiken.
    a. Ontleden
    b. Abstractie
    c. Algoritmisch denken (code)
    d. Evalueren
    e. Veralgemeniseren
7. Argumenteren op basis van bewijzen.
8. Informatie verkrijgen, beoordelen en doorgeven.

Wat is computational thinking?

De term “computational thinking” werd als eerste gebruikt door Seymour Papert, maar kreeg echte bekendheid door professor Jeannette Wing. Zij definieerde computational thinking als volgt:

”de denkprocessen waarmee problemen en hun oplossingen zodanig worden geformuleerd dat ze kunnen worden gepresenteerd in een vorm die effectief kan worden uitgevoerd door een informatieverwerkende tussenpersoon.” (Wing, 2011)

Computational thinking wordt op diverse gebieden, in verschillende situaties en in het dagelijks leven gebruikt. Je vindt computational thinking-vaardigheden terug in de wetenschap, techniek, wiskunde en informatica. De onderstaande vaardigheden zijn computational thinking-vaardigheden:

Ontleden

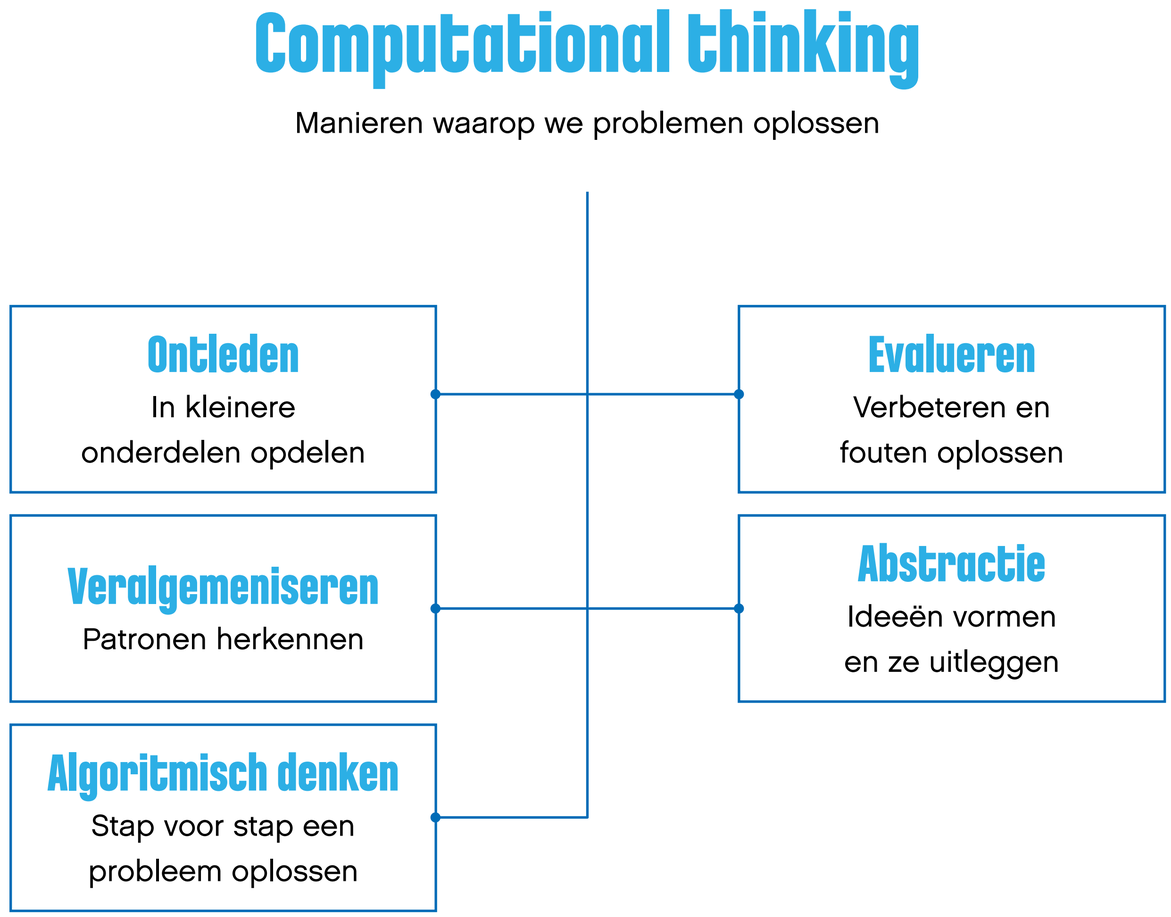
Ontleden is de vaardigheid een probleem in kleinere onderdelen op te delen om gemakkelijker een oplossing te vinden. Op die manier wordt het gemakkelijker het probleem aan iemand anders uit te leggen of het in taken op te delen. Ontleden leidt vaak tot veralgemeniseren.

Voorbeeld: Wanneer je op vakantie gaat, kan de voorbereiding (of het project) in subtaken worden opgedeeld: een vlucht boeken, een hotel reserveren, je koffers inpakken, enz.

Veralgemeniseren (patroonherkenning)

Veralgemeniseren is de vaardigheid om bekende delen van een taak te herkennen, evenals delen die ergens anders in voorkwamen. Dit leidt vaak tot eenvoudigere manieren om algoritmen te creëren.

Voorbeeld: Verkeerslichten werken door het eindeloos herhalen van dezelfde reeks acties.



Wat is computational thinking?

Algoritmisch denken

Algoritmisch denken is de vaardigheid om een geordende reeks stappen te creëren met als doel een probleem op te lossen.

Voorbeeld één: als we iets koken aan de hand van een recept, volgen we een reeks stappen om een maaltijd te bereiden.
Voorbeeld twee: als we met een computer werken, kunnen we een opeenvolging van handelingen programmeren die de computer opdraagt wat hij moet doen.

Evalueren of fouten oplossen

Dit is de vaardigheid om na te gaan of een proefmodel al dan niet werkt zoals je had bedoeld en zo niet, de vaardigheid om te bepalen wat moet worden verbeterd. Dit is ook de procedure die een computerprogrammeur volgt om fouten in een programma te zoeken en deze op te lossen.

Voorbeeld één: tijdens het koken proeven we regelmatig van het gerecht om na te gaan of het genoeg gekruid is.
Voorbeeld twee: als we een tekst controleren op spelfouten en ontbrekende leestekens, lossen we de fouten op zodat de tekst weer klopt.

Abstractie

Abstractie is de vaardigheid om een probleem of een oplossing uit te leggen en de onbelangrijke details achterwege te laten. Met andere woorden: in staat zijn een idee te conceptualiseren.

Voorbeeld: Als we een fiets beschrijven, doen we dit aan de hand van enkele details. We vermelden bijvoorbeeld het type en de kleur, en eventueel meer details voor iemand die echt geïnteresseerd is in fietsen.

Een proces voor de ontwikkeling van computational thinking-vaardigheden

Een technisch ontwerpproces gebruiken

Als technici oplossingen voor een probleem zoeken, maken ze gebruik van een ontwerpproces. Ze doorlopen verschillende fasen die hen naar een oplossing leiden. Tijdens elk van deze fasen passen de leerlingen een aantal van hun vaardigheden toe of ontwikkelen ze deze. Deze vaardigheden noemen we “computational thinking-vaardigheden”.

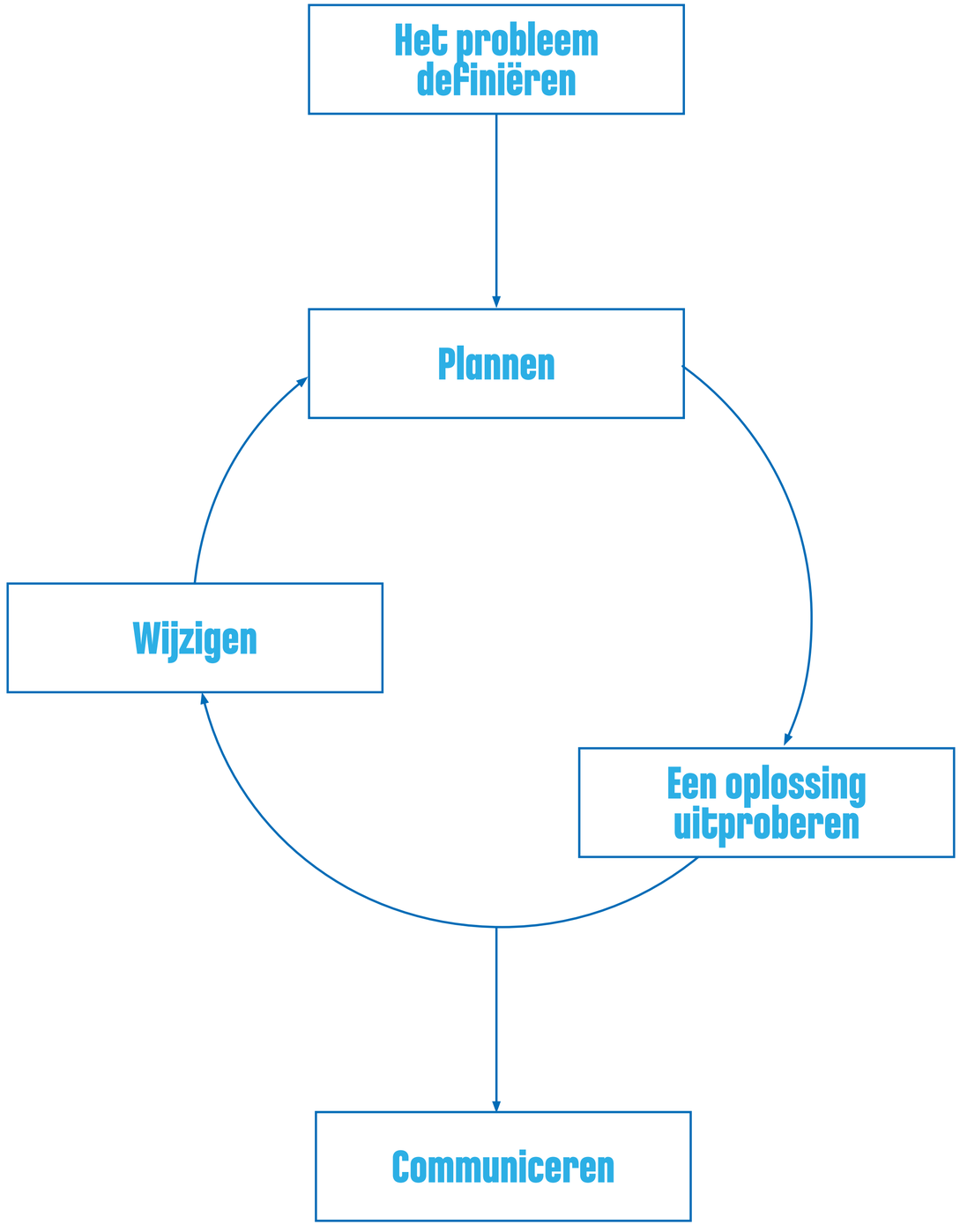
In WeDo 2.0 volgen leerlingen een soortgelijke procedure:

Het probleem definiëren

De leerlingen krijgen een onderwerp aangeboden met een probleem dat ze willen oplossen of een situatie die ze willen verbeteren. Soms bevat een probleem veel details. Het probleem kan dan in kleinere onderdelen worden opgedeeld om gemakkelijker een oplossing te vinden.

Door het probleem op een eenvoudige manier te definiëren en door een aantal criteria voor succes te bepalen, ontwikkelen de leerlingen een vaardigheid die “Ontleden” wordt genoemd.

Met andere woorden:
- *Is de leerling in staat om het probleem zelf uit te leggen?*
- *Is de leerling in staat om te beschrijven op welke manier ze gaan evalueren, ongeacht of ze erin zijn geslaagd het probleem op te lossen?*
- *Is de leerling in staat om het probleem in kleinere en beter beheersbare onderdelen op te delen?*



Een proces voor de ontwikkeling van computational thinking-vaardigheden

Plannen

De leerlingen moeten de tijd nemen om verschillende oplossingen voor het probleem te bedenken en vervolgens een gedetailleerd plan maken om één van hun ideeën uit te voeren. Ze bepalen zelf de stappen die ze moeten doorlopen om de oplossing te vinden. Door te bepalen welke onderdelen van de taak ze misschien al eerder hebben gezien, ontwikkelen ze een vaardigheid die “Veralgemeniseren” wordt genoemd.

Met andere woorden:
- *Is de leerling in staat om een lijst met acties op te stellen voor het programmeren?*
- *Is de leerling in staat om onderdelen van programma‘s te herkennen die hij of zij zou kunnen gebruiken?*
- *Is de leerling in staat om onderdelen van programma‘s opnieuw te gebruiken?*

Uitproberen

Elke leerling krijgt daarna de opdracht de eindversie van zijn of haar oplossing te creëren. In deze fase van het proces gebruiken leerlingen een programmeertaal met pictogrammen om hun LEGO® modellen te activeren. Wanneer de leerlingen hun ideeën coderen, ontwikkelen ze hun vaardigheden voor algoritmisch denken.

Met andere woorden:
- *Is de leerling in staat een oplossing voor een programma te programmeren?*
- *Is de leerling in staat om opeenvolgingen, herhalingen, voorwaarden, enz. te gebruiken?*

Wijzigen

De leerlingen evalueren hun oplossing en gaan na of hun programma en model al dan niet aan de criteria voor succes voldoen. Door hun evaluatievaardigheden te gebruiken, bepalen ze of ze een deel van hun programma moeten veranderen, herstellen of verbeteren of dat ze er fouten uit moeten halen.

Met andere woorden:
- *Gebruikt de leerling delen van het programma opnieuw?*
- *Lost de leerling problemen in het programma op?*
- *Is de leerling in staat om te bepalen of de oplossing verband houdt met het probleem?*

Een proces voor de ontwikkeling van computational thinking-vaardigheden

Communiceren
De leerlingen presenteren de eindversie van hun oplossing voor de klas en leggen hierbij uit waarom hun oplossing aan de criteria voor succes voldoet. Door hun oplossing op het juiste detailniveau uit te leggen, ontwikkelen ze hun abstractie- en communicatievaardigheden.

Met andere woorden:
- Legt de *leerling het belangrijkste deel van de oplossing uit?*
- *Geeft de leerling genoeg details om de begrijpelijkheid te vergroten?*
- *Zorgt de leerling ervoor dat deze uitlegt hoe de oplossing aan de succesvereisten voldoet*?

Vaardigheden voor computational thinking ontwikkelen via coderen

De leerlingen maken kennis met een aantal programmeerprincipes om hun algoritmisch denken te stimuleren. Tijdens het ontwerpen van hun oplossingen zetten ze een reeks acties en structuren op waarmee hun modellen tot leven komen.

De meest voorkomende WeDo 2.0-programmeerprincipes die de leerlingen gaan gebruiken zijn:

1. Output
Output is alles wat wordt bestuurd door het programma dat de leerlingen schrijven. Voorbeelden van output in WeDo 2.0 zijn geluiden, verlichting, beeld en het in- en uitschakelen van motoren.

2. Input
Input is de informatie die een computer of apparaat ontvangt. Deze informatie kan door het gebruik van sensoren in de vorm van een numerieke waarde of een tekst worden ingevoerd. Een sensor die bijvoorbeeld iets waarneemt of meet (zoals een afstand) zet die waarde om in een digitaal inputsignaal zodat het in een programma kan worden gebruikt.

3. (Wachten op) gebeurtenissen
De leerlingen kunnen hun programma laten weten dat er moet worden gewacht tot er iets gebeurt voordat het de opeenvolging van acties hervat. Programma's kunnen gedurende een vooraf ingestelde tijd wachten of wachten tot een sensor iets detecteert.

4. Herhaling
De leerlingen kunnen programmeren dat de acties ofwel eindeloos, ofwel voor een bepaalde duur worden herhaald.

5. Functies
Functies zijn een groep acties die in specifieke situaties samen moeten worden gebruikt.
Bijvoorbeeld: de groep stenen die kan worden gebruikt om een licht te laten knipperen, zou de knipperfunctie kunnen worden genoemd.

6. Voorwaarden
Voorwaarden worden door de leerlingen gebruikt voor het programmeren van acties die enkel onder bepaalde omstandigheden moeten worden uitgevoerd. Het creëren van voorwaarden in een programma betekent dat een deel van het programma nooit wordt uitgevoerd als niet aan de voorwaarde wordt voldaan. Bijvoorbeeld: als de kantelsensor naar links kantelt, start de motor en als de sensor naar rechts kantelt, stopt de motor. Als de kantelsensor nooit naar links kantelt, dan start de motor nooit en als de sensor nooit naar rechts kantelt, dan stopt de motor nooit.