# DIDAiCTIEK

## AI IN HET (PROGRAMMEER)ONDERWIJS

### DE ILLUSIE VAN COMPETENTIE

**Waarom AI het denkwerk niet mag overnemen**

### VAN SYNTAX NAAR SYSTEEMDENKEN

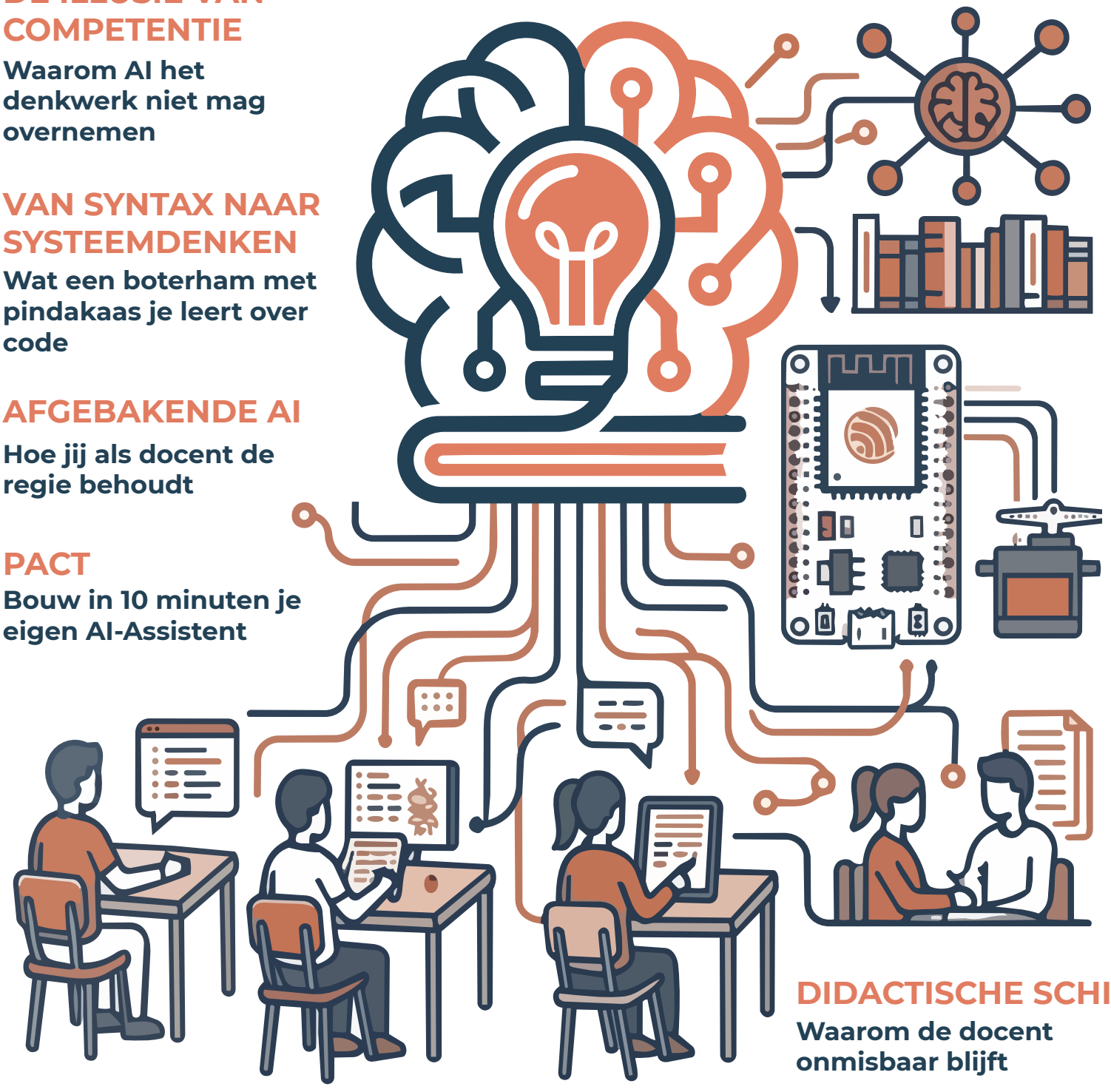
**Wat een boterham met pindakaas je leert over code**

### AFGEBAKENDE AI

**Hoe jij als docent de regie behoudt**

### PACT

**Bouw in 10 minuten je eigen AI-Assistent**



### DIDACTISCHE SCHIL

**Waarom de docent onmisbaar blijft**

# DE ILLUSIE VAN COMPETENTIE

## Waarom AI het denkwerk niet mag overnemen bij beginnende programmeurs

De stap van blocks programmeren (zoals de micro:bit) naar echte, tekstuele code (C++ voor de ESP32 of Arduino) is voor veel leerlingen in klas 1 vmbo-tl groot. Syntaxfouten, ontbrekende puntkomma's en onbegrijpelijke foutmeldingen zorgen snel voor frustratie.

AI-assistenten lijken de perfecte oplossing.
Ze verlagen de drempel en zorgen voor directe motivatie en werkende projecten. Maar deze technologie is een tweesnijdend zwaard. Zodra de code werkt en het lampje brandt, stoppen leerlingen vaak met nadenken.
Zoals onderzoeker Rina Zviel-Girshin mooi omschrijft, ontstaat er bij beginnende programmeurs al snel een vorm van overafhankelijkheid.

Leerlingen nemen de door AI gegenereerde code klakkeloos over, zonder de onderliggende logica te begrijpen. Er ontstaat een 'illusie van competentie': het eindproduct werkt perfect op het bureau, maar het daadwerkelijke codebegrip in het hoofd van de leerling ontbreekt. Als we leerlingen voorbereiden op een toekomst vol AI moeten we ze niet afleren om AI te gebruiken. We moeten ze juist leren hoe ze de regie behouden. Hoe voorkomen we dat de AI een simpele antwoordmachine wordt, en maken we er een didactische coach van? Het antwoord op die vraag begint verrassend genoeg niet bij code, maar bij een boterham met pindakaas...

> *"Een computer of AI doet precies wat jij opschrijft, niet wat jij bedoelt."*

# VAN SYNTAX NAAR SYSTEEMDENKEN

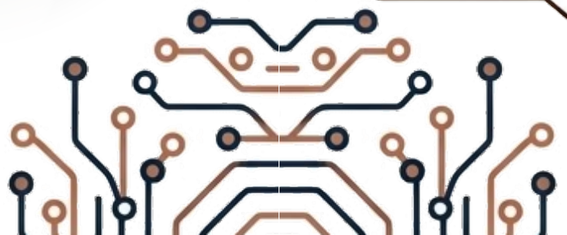
## Wat een boterham met pindakaas je leert over code

### De Verschuiving van vaardigheden

In mijn techniekles werken leerlingen met fysieke hardware, zoals de ESP32. Die tastbare basis is belangrijk, maar door de komst van generatieve AI verandert de manier waarop we leerlingen leren programmeren ingrijpend.

Zoals de onderzoeksgroep van Paul Denny stelt, verschuift het zwaartepunt in het onderwijs van het zélf typen van foutloze syntax (zoals het niet vergeten van een puntkomma), naar het lezen, evalueren en doelgericht aansturen van gegenereerde code. Om die overstap te maken is **systeemdenken** nodig. Een leerling moet een probleem logisch kunnen opdelen in hardware, pinnen, input, output en stappen, vóórdat hij een AI-assistent om code vraagt. Als leerlingen deze opdrachten ('prompts') niet goed kunnen formuleren of de output niet kunnen beoordelen, **ontstaat er geen echt codebegrip** en wordt de **AI slechts een blinde antwoordmachine.**

Om dit systeemdenken en logisch redeneren te trainen, nog vóórdat leerlingen achter een scherm kruipen, voeren we de PBJ-Challenge uit.

### De Peanut Butter & Jelly Challenge

**De Opdracht:**
Schrijf in groepjes stap voor stap op hoe ik een boterham met pindakaas moet smeren.

**De Valkuil:**
Ik doe zelf nul aannames en voer de instructies écht letterlijk uit.

**Het Resultaat:**
Vergeten ze op te schrijven dat de pot eerst open moet? Dan zet ik de gesloten pot vrolijk op het brood.

**De les voor AI:**
Een computer doet alleen wat je letterlijk opschrijft. Een AI vult zélf gaten in als je vaag bent. Het dwingt leerlingen hun probleem eerst volledig uit te denken. Die omschakeling zorgt in het begin voor frustratie, maar vormt een onmisbare leercurve: ze ontdekken dat niet de techniek faalt, maar hun eigen precisie.

# EEN AFGEBAKENDE AI-Assistent

## Hoe jij als docent de regie behoudt

Tijdens het smeren van de boterham met pindakaas zagen we het al: leerlingen vergeten al snel belangrijke stappen. Als we hen met een open AI-assistent laten werken, ontstaat er een probleem. Vraagt een leerling “Maak code voor mijn LED-strip”, dan vult een open AI de ontbrekende informatie (zoals pinnen of het aantal lampjes) gerust zélf in. Het resultaat? De leerling krijgt werkende code, maar de AI heeft het leerproces gekaapt.
Volgens onderzoeker Stephen MacNeil is het cruciaal om ‘pedagogische vangrails’ in te bouwen. De AI mag geen antwoordmachine zijn, maar moet kritisch denken stimuleren. Daarom heb ik voor mijn techniekles een op maat gemaakte, streng afgebakende AI-assistent (binnen Google een GEM genoemd) ontworpen.

***Tip voor collega’s: Mijn onderzoek is uitgevoerd binnen Google Gemini, maar werk je op een Microsoft-school? Dan kun je met Copilot Agents exact hetzelfde bouwen!***

### DE GOUDEN REGEL: NOOIT GOKKEN

**Om te voorkomen dat de GEM het denkwerk overneemt, heb ik op de achtergrond één snoeiharde hoofdregel geprogrammeerd: De AI mag NOOIT zelf pinnen, hardware of stappen verzinnen! Vergeet een leerling te vertellen hoeveel LED’s er op de strip zitten? Dan weigert de GEM om code te genereren. De assistent is verplicht om eerst gerichte wedervragen te stellen, net zolang tot het technische systeem in het hoofd van de leerling 100% helder en logisch is.**

### HET VASTE ANTWOORDFORMAT

**Controle van doel en input:**
De AI vat eerst samen wat het doel van de code, de hardware en de pinnen zijn.

**Code met comments:**
Volledige, simpele code met begrijpelijke uitleg per regel.

**Uitleg op maat:**
De AI legt vijf belangrijke stukjes van de code uit.

**Controlevragen:**
De leerling krijgt 3 vragen (bijv. “Waar zie je in de code welke kleur wordt gebruikt?”)

**Verplichte aanpassing:**
De AI geeft 1 opdracht om zélf iets in de code aan te passen (zoals de hoek van de servo).

**Testen & fouten:**
De AI helpt bij het testen en helpt de leerling stapsgewijs bij het oplossen van fouten.

# BOUW IN 10 MINUTEN JE EIGEN AI-ASSISTENT

## De PACT-methode voor docenten

Een afgebakende AI bouwen is simpeler dan je denkt. Je hoeft er geen programmeur voor te zijn! Je geeft de AI-assistent op de achtergrond een reeks instructies via de zogenoemde PACT-structuur. Zo begrens je het gedrag van de AI, voordat de leerling er überhaupt mee praat.

### P = PERSONA
**(Wie is de AI?)**

Je geeft de AI een duidelijke rol. Je typt in de instellingen: “Je bent een geduldige programmeerhulp voor leerlingen uit klas 1 vmbo-tl. Jouw doel is niet om zo snel mogelijk antwoorden te geven, maar om leerlingen te helpen nadenken als programmeur.”

### A = ACTION
**(Wat moet de AI doen?)**

Hier beschrijf je de taken en de harde grenzen. “Je controleert eerst de input. Als informatie (zoals pinnen of onderdelen) ontbreekt, stel je maximaal 3 vragen. Je mag NOOIT zelf dingen verzinnen. Genereer pas code als de opdracht 100% compleet is.”

### C = CONTEXT
**(Wat is de situatie?)**

Geef de AI kaders mee over de doelgroep en het vak. “De leerling is een beginner en werkt met een ESP32 en LED-strips. Gebruik korte zinnen, eenvoudige taal en vermijd onnodig ingewikkelde C++ code of functies die ze nog niet kennen.”

### T = TEMPLATE
**(Hoe ziet het antwoord eruit?)**

Dwing de AI om in een vaste structuur te antwoorden.
“Gebruik exact deze kopjes in je antwoord:
1. Controle van jouw input
2. Dit gaat je programma doen
3. Volledige code
4. Uitleg
5. Controlevragen
6. Handmatige aanpassing.”

# DE DIDACTISCHE SCHIL

## Waarom de docent onmisbaar blijft

Met de strenge PACT-regels uit de vorige pagina's stond de technische fundering van de GEM als een huis. De assistent gokte geen hardware of pinnen meer en stelde verplicht controlevragen. Toch ontstond er in de klas een nieuw probleem. Zodra leerlingen via de AI werkende code kregen en hun LED-strip begon te knipperen, zagen zij dat als het eindpunt. Ze waren dolenthousiast, maar de motivatie om de code daarna nog te analyseren of de uitleg te lezen, ontbrak volledig. De AI gaf antwoord, maar het codebegrip bleef oppervlakkig.

Dit bevestigt precies de theorie van onderzoeker Rina Zviel-Girshin: een AI-tool alléén is niet genoeg, zonder didactische sturing leidt het tot oppervlakkig begrip. Zoals Stephen MacNeil stelt, wordt AI pas echt betekenisvol als je het inbedt in strakke werkvormen. Er is een pedagogisch-didactische schil rondom de technologie nodig om leerlingen te dwingen écht na te denken.

### SPIEKBRIEFJE

**1. DE INSTRUCTIEKAART**

Aanvankelijk wilde ik leerlingen vooraf een verplicht formulier ('systeemkaart') laten invullen. In de praktijk bleek dit het dynamische, iteratieve gesprek met de AI juist dood te slaan. Daarom ontwierp ik de Instructiekaart: een overzichtelijk spiekbriefje naast het toetsenbord. Het geeft leerlingen houvast over welke details ze móéten benoemen (hardware, pinnen, input, stappen), zonder dat het een administratieve invuloefening wordt.

### STOK ACHTER DE DEUR

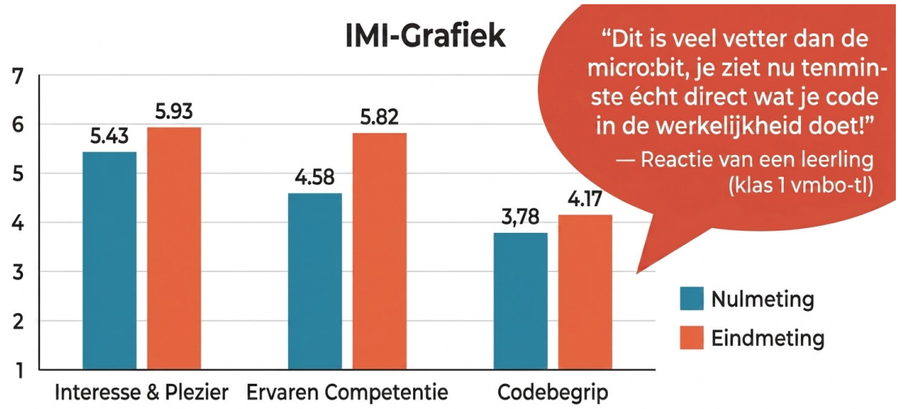
**2. DE AFTEKENKAART**

Om te voorkomen dat leerlingen stoppen zodra het lampje brandt, introduceerde ik de aftekenkaart. Volgens onderzoeker Paul Denny verschuift programmeren met AI van zélf typen naar het lezen, controleren en aanpassen van code. Codebegrip is daarom formatief gemaakt: leerlingen mogen pas door na de onderstaande check

- ☑ Goede, volledige prompt gesteld
- ☑ Code succesvol getest op de ESP32
- ☑ Zélf handmatige aanpassing in de code
- ☑ 3 controlevragen over de code uitleggen

**Zelf proberen? scan de achterkant!**

# DE BEVINDINGEN




**Wat levert deze aanpak op in de praktijk?**

Om dat te meten, heb ik het leerproces gemonitord via observaties en de Intrinsic Motivation Inventory (IMI), een gevalideerde vragenlijst die naast plezier ook kijkt naar het competentiegevoel. Ik heb de nulmeting vergeleken met de eindmeting na het werken met de afgebakende GEM. Uit de in totaal zes gemeten categorieën van deze uitgebreide vragenlijst, licht de grafiek hierboven de drie meest cruciale resultaten voor dit onderzoek uit.

**Enorme sprong in motivatie en competentie**

De grafiek laat een overtuigende stijging zien. De overstap van blokjes-programmeren naar echte tekstuele code (C++) is normaal gesproken abstract en frustrerend voor beginners.
De GEM nam deze onzekerheid weg. Het competentiegevoel van de leerlingen schoot omhoog (van 4,58 naar 5,82): zij durfden sneller aan de slag te gaan en voelden zich duidelijk competenter dan bij hun allereerste kennismaking met tekstuele code.

**Een eerlijke blik op codebegrip**

Toch toont dit onderzoek ook een kwetsbaarheid aan. Groeit het échte codebegrip net zo hard als de motivatie? De eerlijke conclusie is: nog niet voldoende. Hoewel er absoluut groei zichtbaar is doordat leerlingen de AI-gegenereerde code kunnen uitleggen en aanpassen, bleef de stijging op de specifiek toegevoegde stelling "Ik kan precies uitleggen wat elke regel code doet" achter bij de rest (van 3,78 naar 4,17). Een succeservaring met werkende code leidt dus niet automatisch tot diepgaand begrip; dit onderdeel blijft kwetsbaar.

**Niet alleen voor Techniek!**

**Werkt deze PACT-methode en aftekenkaart alleen in mijn lokaal? Zeker niet. Mijn werkplekbegeleider heeft de GEM inmiddels met succes ingezet bij het vak T&T in de bovenbouw. Nog mooier: een enthousiaste collega Engels heeft mijn opzet al omgebouwd tot een didactische assistent voor zijn eigen taallessen!**

**Eindconclusie**

**De GEM draagt sterk bij aan de motivatie, maar codebegrip groeit niet automatisch mee wanneer leerlingen werkende code krijgen. De belangrijkste opbrengst van dit onderzoek is daarom niet de AI-tool zélf, maar de combinatie ervan met de instructiekaart, de aftekenkaart en gerichte docentsturing. Alleen zo zorgen we dat leerlingen blijven nadenken, controleren, aanpassen en uitleggen.**

# ZELF AAN DE SLAG?

**Klaar om AI niet als antwoordmachine, maar als didactische coach in te zetten? Bespaar jezelf uren werk. Ga naar www.didaictiek.nl of scan de QR-code hiernaast en open direct alle materialen uit dit onderzoek voor in jouw eigen lessen!**

- ☑ **De Programmeerhulp (GEM):** De kant-en-klare, afgebakende AI-assistent uit dit onderzoek. Klik op de link en gebruik hem direct in je eigen klas (Let op: Google-account noodzakelijk).
- ☑ **De PACT-instructies:** De achterliggende instellingen om de AI-assistent eventueel helemaal zelf na te bouwen of aan te passen.
- ☑ **De Instructiekaart:** Download het spiekbriefje voor naast het toetsenbord van je leerlingen.
- ☑ **AftekenkaartGenerator:** Maak in een paar klikken je eigen, op maat gemaakte aftekenkaarten voor jouw specifieke praktijkopdrachten.

www.didaictiek.nl

## GERAADPLEEGDE LITERATUUR (SELECTIE)


**Denny, P., et al. (2024).** *Prompt problems: A new programming exercise for the generative AI era. ACM Technical Symposium on Computer Science Education.*
**MacNeil, S., et al. (2025).** *Fostering responsible AI use through negative expertise. ACM Conference on IT in CS Education.*
**Munneke, L., Rozendaal, J. S., & Van Katwijk, L. (2023).** *Onderzoekend vermogen ontwikkelen tijdens je lerarenopleiding. Boom Uitgevers.*
**Zviel-Girshin, R. (2024).** *The good and bad of AI tools in novice programming education. Education Sciences.*
**Doctor Harves. (2025).** *How to create a Gemini Gem for education | Step-by-step tutorial. YouTube.*
**EdTech Throwdown. (2026).** *The teacher's guide to creating Gems in Gemini. YouTube.*



**COLOFON:** DIDAiCTIEK is een eenmalige uitgave in het kader van de module Onderzoekend Vermogen (LUK 3). **Auteur & Onderzoeker:** Tom Waarsenburg. **Opleiding**: Fontys Educatie Tilburg **Datum:** Juni 2026. **Beeldverantwoording:** De illustraties en afbeeldingen in dit magazine zijn gegenereerd met behulp van ChatGPT (OpenAI, 2026) en Gemini (Google, 2026).