Onderzoek verslag

Auteur: Diyar Ali
Studentnummer: 500924691
E-mail: diyar.ali@hva.nl
Telefoonnummer: 06 85154358
Plaats en datum: Amsterdam, 8 mei 2024
Onderwijsinstelling: Hogeschool van Amsterdam

Opleiding en leerroute: HBO-ICT, Software Engineering
Begeleidende docent: Stefan de Kraker

Voorwoord

Ter afronding van mijn propedeuse jaar in de vierjarige bacheloropleiding HBO-ICT richting Software Engineering aan de Hogeschool van Amsterdam, heb ik onderzoek gedaan naar een belangrijk aspect van programmeeronderwijs voor leerlingen in de onderbouw van het voortgezet onderwijs. Mijn focus lag op het identificeren van een geschikte programmeertaal.

Mijn keuze voor deze onderzoeksvraag is voortgekomen uit mijn ervaring als assistent bij het geven van programmeer- en ontwerp lessen aan leerlingen in de onderbouw van het voortgezet onderwijs. Mijn rol hierbij is het ondersteunen bij het aanleren van programmeervaardigheden, met als specifiek doel de basisprincipes van probleemformulering en oplossingsstrategieën over te brengen. Aangezien programmeren voor deze leerlingen nieuw is, is er een dringende behoefte aan een gestructureerd en doelgericht curriculum. De keuze van de juiste programmeertaal is daarbij cruciaal en heeft geleid tot de centrale vraag van dit onderzoek: "Welke programmeertaal is het meest geschikt om leerlingen in de onderbouw van het voortgezet onderwijs de basis van probleemformulering en oplossingsstrategieën aan te leren?"

Tot slot wil ik in dit voorwoord mijn oprechte dank uitspreken aan alle betrokkenen die hebben bijgedragen aan de totstandkoming van dit onderzoek. Mijn dank gaat uit naar mijn begeleiders en docenten voor hun waardevolle begeleiding en feedback gedurende dit proces.

Samenvatting

Dit onderzoek richt zich op het identificeren van de meest geschikte programmeertaal voor het onderwijzen van probleemformulering en oplossingsstrategieën aan leerlingen in de onderbouw van het voortgezet onderwijs. Door middel van een uitgebreide literatuurstudie en analyse van verschillende programmeerparadigma's wordt een aanbeveling gedaan voor een taal die zowel effectief als toegankelijk is voor deze doelgroep.

Volgens eerdere onderzoeken en bronnen variëren de specifieke behoeften en capaciteiten van leerlingen met betrekking tot het aanleren van probleemformulering en oplossingsstrategieën per individu, afhankelijk van hun cognitieve gereedheid en bereidheid. Leerlingen met een sterk ontwikkeld kritisch denkvermogen moeten worden gestimuleerd met morele en academische vraagstukken die voor hen emotionele waarde hebben. Deze vraagstukken moeten worden verspreid over een breed scala van vakken, afhankelijk van de leerdoelen, interesses en probleemstellingen van de leerling.

Voor het onderwijzen van kinderen zijn visuele en eenvoudige programmeertalen vaak het meest effectief. Een blok gebaseerde, visuele programmeertaal die elementen van het imperatieve en objectgeoriënteerde paradigma combineert, is bijzonder geschikt. Naarmate leerlingen meer ervaring opdoen, kunnen zij worden geïntroduceerd in complexere paradigma's zoals objectgeoriënteerd en functioneel programmeren, afhankelijk van hun interesses en vaardigheden.

Inhoud

Inleiding

In dit document worden de resultaten, conclusies en aanbevelingen besproken die zijn voortgekomen uit het onderzoek ter afronding van het propedeusejaar in de vierjarige bacheloropleiding HBO-ICT richting Software Engineering aan de Hogeschool van Amsterdam.

Probleemschets

Als een veeltalige heb ik altijd een fascinatie gehad voor hoe we taal gebruiken en hoe dat onze gedachten hervormt. Ik heb mijn jonge jaren in Syrië doorgebracht waarbij ik Koerdisch en Arabisch sprak, daarna verhuisde ik naar Nederland op een leeftijd van 14 jaar. Ik sprak al een beetje Engels voor het emigreren, maar ik kan me de verschillen nog goed herinneren, niet alleen in woorden die we gebruiken in verschillende talen maar de subtiele verschillen in betekenis en connotatie van die woorden. Het feit dat ik steeds meer begon te denken in de taal van mijn nieuwe thuis leek het belangrijkste element van mijn integratie in het Nederlandse leven.

In zijn “Course in General Linguistics" schreef de Zwitserse theoreticus en denker Ferdinand de Saussure over de dynamische en symbiotische relatie tussen een taal en de cultuur die daaruit voortkomt. Dit idee - linguistic relativity - was verder ontwikkeld naar de Sapir-Whorf hypothese, die beweerde dat een groot deel van iemands kijk- en wereldbeeld kan worden verklaard door de taal die hij spreekt.

In mijn huidige omgeving komt het vaak voor dat de taal die mijn teamleden het beste beheersen niet gesproken wordt, maar gecodeerd. Ik ben geïnteresseerd in het idee dat de Sapir-Whorf-hypothese kan worden toegepast op programmeertalen. Programmeertalen zijn instrumenten, elk ontworpen om met behulp van een computer specifieke problemen op te lossen. Of het nu Python, C, Scratch of COBOL is, programmeertalen zijn gebaseerd op bepaalde paradigma's die elk hun eigen culturen, houdingen en neigingen hebben. Een groot deel van iemands kijk- en wereldbeeld, met name in het praktische domein zoals het analyseren, definiëren, evalueren en toepassen van oplossingsstrategieën, kan worden beïnvloed door de programmeertaal die hij beheerst.

Leeswijzer

Dit onderzoek is gestructureerd in verschillende secties, elk met een specifiek doel om de lezer door de bevindingen te leiden. Hieronder volgt een overzicht van de inhoud en wat u in elk gedeelte kunt verwachten:

1. Voorwoord
   - Het voorwoord introduceert het onderzoek en de context ervan. Hierin worden de aanleiding, de persoonlijke motivatie van de auteur, en de dankbetuigingen aan degenen die hebben bijgedragen aan het onderzoek besproken.

2. Samenvatting
   - De samenvatting biedt een beknopt overzicht van de onderzoeksvraag, de methodiek, de belangrijkste bevindingen en de conclusies. Dit gedeelte geeft de lezer een snel inzicht in de inhoud van het rapport.

3. Inleiding
   - De inleiding beschrijft de achtergrond van het onderzoek, de probleemstelling, en de relevantie ervan. Het biedt een contextuele basis voor het onderzoek

4. Methodiek
   - In de methodiek wordt gedetailleerd uitgelegd welke onderzoeksmethoden zijn gebruikt. Hier wordt ook de keuze voor deze methoden en hun relevantie voor het onderzoek uiteengezet.

5. Resultaten en Discussie
   - Dit gedeelte presenteert de bevindingen van het onderzoek. Het bevat een analyse van de verschillende programmeerparadigma's en hun geschiktheid voor het onderwijzen van probleemformulering en oplossingsstrategieën aan leerlingen in de onderbouw van het voortgezet onderwijs. De discussie interpreteert deze resultaten en vergelijkt ze met de bestaande literatuur.

6. Conclusies en Aanbevelingen
   - De conclusies geven een samenvatting van de belangrijkste bevindingen en beantwoorden de onderzoeksvraag.

Overzicht Hoofdstukken

Hoofdstuk 1: Specifieke Behoeften en Capaciteiten van Leerlingen
- Dit hoofdstuk onderzoekt de cognitieve ontwikkeling van leerlingen in de onderbouw van het voortgezet onderwijs. Het bespreekt factoren zoals cognitieve gereedheid, culturele en gendergerelateerde invloeden, en de behoefte aan onderwijs dat hogere-orde denkvaardigheden stimuleert.

Hoofdstuk 2: Geschikte Programmeerparadigma's
- Dit hoofdstuk analyseert verschillende programmeerparadigma's zoals het imperatieve, objectgeoriënteerde, functionele, logische en blokgebaseerde paradigma. Het beoordeelt hun kenmerken, gebruik, en geschiktheid voor het onderwijs aan kinderen.

Methodiek

Voor dit onderzoek wordt gebruikgemaakt van twee benaderingen:

1. Bronnenonderzoek (literatuurstudie): Hierbij wordt een verzameling gemaakt van schriftelijke informatie uit diverse bronnen, waaronder (wetenschappelijke) tijdschriften, boeken, rapporten, verslagen, dagboeken, enzovoort. Deze methode stelt ons in staat om een breed scala aan bestaande kennis en inzichten te verzamelen en te analyseren.

2. Desk research: Hierbij wordt gebruikgemaakt van bestaande gegevens en literatuur om informatie te analyseren en te synthetiseren. Dit type onderzoek is gericht op het analyseren van beschikbare kennis en het trekken van conclusies zonder nieuwe data te verzamelen.

Uitleg

Gezien de aard van het onderzoek en de richtlijnen van het hoger beroepsonderwijs, is een praktische benadering van cruciaal belang.
Dit onderzoek richt zich op het verkrijgen van inzicht in de behoeften en capaciteiten van leerlingen in de onderbouw van het voortgezet onderwijs, evenals de theoretische benaderingen die het meest geschikt zijn voor hun leerproces. Een literatuurstudie biedt de meest geschikte methode om deze informatie te verzamelen, omdat het ons in staat stelt om kritisch te evalueren wat er al bekend is over de behoeften van leerlingen.

Daarnaast wordt een deskresearch uitgevoerd om de effectiviteit van verschillende programmeerparadigma's en -talen in het onderwijs te analyseren en te synthetiseren. Deze analyse richt zich op het gebruik van programmeertalen voor het aanleren van probleemoplossend denken aan leerlingen in de onderbouw van het voortgezet onderwijs. Op basis van deze analyse zullen aanbevelingen worden gedaan voor het gebruik van programmeertalen in het onderwijs.

Onderzoeksvragen

Hoofdvraag:
Welke programmeertaal is het meest geschikt om leerlingen in de onderbouw van het voortgezet onderwijs de basis van probleemformulering en oplossingsstrategieën aan te leren.

Deelvragen:
Wat zijn de specifieke behoeften en capaciteiten van leerlingen in de onderbouw van het voortgezet onderwijs met betrekking tot het aanleren van probleemformulering en oplossingsstrategieën?

Volgens Piaget (Jean Piaget's theory of cognitive development) als een kind 11 jaar of ouder is komt die in een "formal operational stage". Dit stadia houdt in dat als adolescenten deze fase ingaan, verwerven ze het vermogen om op een abstracte manier te denken, het vermogen om items op een meer verfijnde manier te combineren en te classificeren, en het vermogen om op een hogere orde te redeneren.
Het potentieel van iemands vermogen staat niet gelijk aan het vermogen zelf. Zo betekent het vermogen om op een hoger niveau te redeneren niet automatisch dat men daadwerkelijk op een hoger niveau kan redeneren. Dit vermogen moet verder ontwikkeld worden, en daarvoor is het nodig om te begrijpen wat deze ontwikkeling stimuleert en wat deze belemmert.

Welk programmeerparadigma biedt de meest geschikte theoretische benadering voor het aanleren van probleemformulering en oplossingsstrategieën aan leerlingen in de onderbouw van het voortgezet onderwijs?

Programmeerparadigma's vormen verschillende benaderingen van het structureren en uitvoeren van computerprogramma's, en elk heeft zijn eigen theoretische basis. Door de theoretische basis van elk paradigma te begrijpen, kunnen we ze classificeren binnen de domeinen van computationeel denken. Dit stelt ons in staat om een hiërarchie te demonstreren van de meest geschikte paradigma's voor het onderwijs in de onderbouw van het voortgezet onderwijs, rekening houdend met de behoefte en capaciteiten van leerlingen.

Binnen het gekozen programmeerparadigma, welke programmeertaal is het meest geschikt om te voldoen aan de leerbehoeften van leerlingen in de onderbouw van het voortgezet onderwijs bij het aanleren van probleemformulering en oplossingsstrategieën?

Bij het kiezen van een programmeerparadigma moet er binnen dit paradigma gezocht worden naar de meest geschikte programmeertaal, beoordeeld op basis van criteria die de geschiktheid voor onderwijs bepalen.

Hoofdstuk 1

Dit hoofdstuk onderzoekt de cognitieve ontwikkeling van leerlingen in de onderbouw van het voortgezet onderwijs. Het bespreekt factoren zoals cognitieve gereedheid, culturele en gendergerelateerde invloeden, en de behoefte aan onderwijs dat hogere-orde denkvaardigheden stimuleert.

Wat zijn de specifieke behoeften en capaciteiten van leerlingen in de onderbouw van het voortgezet onderwijs met betrekking tot het aanleren van probleemformulering en oplossingsstrategieën?

De specifieke behoeften en capaciteiten van leerlingen in de onderbouw van het voortgezet onderwijs met betrekking tot het aanleren van probleemformulering en oplossingsstrategieën zijn sterk afhankelijk van de individuele ontwikkeling van de leerling.

Leerlingen in de onderbouw zijn doorgaans tussen de 12 en 15 jaar, met enkele uitzonderingen. Op deze leeftijd is hun cognitieve ontwikkeling, inclusief vaardigheden zoals probleemformulering en het ontwikkelen van oplossingsstrategieën, beïnvloed door culturele en gendergerelateerde factoren. Deze factoren hebben invloed op hun motivatie, leerstijlen en hun houding ten opzichte van leren en succes op school. Zoals onderzoek heeft aangetoond, beïnvloedt de cultuur van jongvolwassenen hun vriendschap patronen (DuBois & Hirsch, 1990) en hun identiteitsontwikkeling (Wisher & Deyhle, 1989). Geslacht speelt eveneens een rol in het zelfbeeld van 10- tot 15-jarigen.

Volgens een onderzoek van het California State Department of Education (1987) beginnen jongvolwassenen op deze leeftijd het vermogen te ontwikkelen om hypothetisch, reflectief en abstract te denken. Dit omvat vaardigheden zoals het analyseren en synthetiseren van gegevens, het stellen en onderzoeken van vragen, het experimenteren, redeneren en het toepassen van verschillende strategieën en oplossingen op problemen.

Individuele Cognitieve Gereedheid
Bij het onderzoeken van de behoeften en capaciteiten van leerlingen moet rekening worden gehouden met de individuele cognitieve gereedheid. Sommige leerlingen zijn klaar voor hogere niveaus van cognitieve activiteiten en moeten daarom worden uitgedaagd met activiteiten die een hogere cognitieve betrokkenheid vereisen. Toepfer (Toepfer model - 1988) moedigt leraren aan om te evalueren of de cognitieve gereedheid van leerlingen hen in staat stelt om deel te nemen aan intensievere intellectuele activiteiten.

Mogelijkheden voor Basis- en Gedeelde Kennis en Geïntegreerde Benaderingen
De verbeterde cognitieve capaciteiten van jongvolwassenen, zoals het vermogen om relaties binnen en tussen academische vakgebieden op te merken, maken het mogelijk om te profiteren van een basis van gemeenschappelijke kennis. Idealiter is deze kennis gebaseerd op de leerbehoeften, interesses en problemen van de leerlingen. Een curriculum dat meerdere domeinen omvat, zoals taal, geschiedenis, kunst, wiskunde, natuurkunde en maatschappijleer, is hierbij wenselijk.

Integratie begint met het toepassen van volledigheid in plaats van het scheiden en fragmenteren van vakken. Ten tweede vindt oprechte integratie in het leerplan plaats

wanneer leerlingen worden geconfronteerd met persoonlijk betekenisvolle vragen en deelnemen aan gelijksoortige ervaringen die ze later kunnen integreren in hun eigen systeem van betekenis (Beane, 1991).

Mogelijkheden voor Hoger-orde Denken

Het vermogen van jongvolwassenen om hypothetisch, reflectief en abstract te denken vraagt om educatieve ervaringen die hogere-orde denkvaardigheden vereisen. Hun snelle ontwikkeling van cognitieve vaardigheden en hun vermogen om na te denken over hun eigen denkprocessen suggereren de behoefte aan complexere gedachten processen, zoals moreel redeneren, kritisch denken en het nemen van weloverwogen beslissingen (California State Department of Education, 1987).

Conclusie

Wat zijn de specifieke behoeften en capaciteiten van leerlingen in de onderbouw van het voortgezet onderwijs met betrekking tot het aanleren van probleemformulering en oplossingsstrategieën? Volgens eerdere onderzoeken en bronnen variëren de specifieke behoeften per kind, afhankelijk van hun individuele bereidheid en cognitieve gereedheid. Leerlingen met een sterk ontwikkeld kritisch denkvermogen moeten worden gestimuleerd door middel van morele en academische vraagstukken die emotionele waarde hebben voor hen. Deze vraagstukken moeten worden verspreid over een breed domein van vakken, afhankelijk van de leerdoelen, interesses en probleemstellingen van de leerling.

Hoofdstuk 2

Welk programmeerparadigma biedt de meest geschikte theoretische benadering voor het aanleren van probleemformulering en oplossingsstrategieën aan leerlingen in de onderbouw van het voortgezet onderwijs?

Een programmeerparadigma is een fundamentele stijl van programmeren die specifieke concepten en methoden benadrukt. In dit overzicht lees je over de meest voorkomende programmeerparadigma's, hun kenmerken, gebruik, en hun geschiktheid om kinderen de basis van probleemformulering en oplossingsstrategieën aan te leren.

1. Imperatief Paradigma

Kenmerken
- Procedures en functies: Programma's worden opgebouwd uit procedures en functies die taken uitvoeren.
- Stapsgewijze instructies: Code bestaat uit opeenvolgende instructies die de computer moet uitvoeren.
- Toestand en mutatie: Er wordt gebruikgemaakt van variabelen die de staat van het programma vertegenwoordigen en die gedurende de uitvoering worden gemuteerd.

Gebruik
- Voorbeelden: C, Pascal, Basic
- Toepassingen: Imperatieve talen zijn geschikt voor algemene programmeer doeleinden, zoals het schrijven van systeemsoftware en applicaties.

Geschiktheid voor kinderen
Imperatieve talen kunnen geschikt zijn voor oudere kinderen die al wat ervaring hebben met programmeren. De stapsgewijze aanpak maakt het gemakkelijk om de logica van een programma te volgen, maar het kan moeilijk zijn voor beginners om complexe *state* transities te begrijpen.
Ook zie je in het onderstaande voorbeeld dat er vrij veel boilerplate-code is, wat een drempel kan vormen voor beginners.

Code Voorbeeld
```c
include <stdio.h>

int main() {
    int i;
    for (i = 0; i < 5; i++) {
        printf("Hello, world!\n");
    }
    return 0;
}
```

2. Objectgeoriënteerd Paradigma

Kenmerken
- Klassen en objecten: Code wordt georganiseerd in klassen en objecten, die gegevens en methoden bevatten.
- Erfenis en polymorfisme: Objecten kunnen eigenschappen en methoden erven van andere klassen, en dezelfde methode kan anders worden geïmplementeerd in verschillende klassen.
- Encapsulatie: Gegevens en methoden worden samen ingekapseld in objecten.

Gebruik
- Voorbeelden: Java, C++, Python
- Toepassingen: Objectgeoriënteerde talen worden vaak gebruikt voor het ontwikkelen van complexe software zoals games, GUI-applicaties en enterprise-software.

Geschiktheid voor kinderen
Objectgeoriënteerde talen zijn geschikt voor kinderen die een basiskennis van programmeren hebben. Het concept van klassen en objecten kan intuïtief zijn, maar de complexiteit van erfenis en polymorfisme kan een uitdaging vormen.

Code Voorbeeld
```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

3. Functioneel Paradigma

Kenmerken
- Pure functies: Functies hebben geen bijwerkingen en geven altijd dezelfde output voor dezelfde input.
- Immutabiliteit: Gegevens zijn onveranderlijk, wat betekent dat ze niet kunnen worden gemuteerd nadat ze zijn gecreëerd.
- Hogere-ordefuncties: Functies kunnen andere functies als argumenten nemen en functies kunnen worden geretourneerd.

Gebruik
- Voorbeelden: Haskell, Lisp, Erlang
- Toepassingen: Functionele talen zijn populair in wetenschappelijk onderzoek, gegevensverwerking en toepassingen waar betrouwbaarheid en voorspelbaarheid belangrijk zijn.

Geschiktheid voor kinderen
Functionele talen zijn over het algemeen minder geschikt voor jonge kinderen of beginners vanwege hun abstracte concepten. Ze kunnen echter waardevol zijn voor oudere kinderen die geïnteresseerd zijn in wiskunde en logica.

Code Voorbeeld
```haskell
main :: IO ()
main = putStrLn "Hello, world!"
```

4. Logisch Paradigma

Kenmerken
- Declaratieve stijl: In plaats van instructies te geven, specificeert de programmeur wat het doel is en de computer zoekt uit hoe het te bereiken.
- Regels en feiten: Programma's bestaan uit een verzameling logische regels en feiten.
- Backtracking en inferentie: De uitvoering omvat het zoeken naar oplossingen door backtracking en inferentie.

Gebruik
- Voorbeelden: Prolog, Datalog
- Toepassingen: Logische talen worden gebruikt in kunstmatige intelligentie, kennisrepresentatie en probleemoplossing.

Geschiktheid voor kinderen
Logische talen zijn meestal niet geschikt voor beginners, maar ze kunnen nuttig zijn voor oudere kinderen die geïnteresseerd zijn in AI en formele logica.

Code Voorbeeld
```prolog
:- initialization(main).

main :-
    write('Hello, world!'), nl.
```

5. Blokgebaseerd Paradigma

Ook blijkt dat block gestructurerde programmeertalen gezien worden als een programmeer paradigma (Goverment Degree College Beerwah)

Kenmerken
- Visuele blokken: Programmeerconcepten worden gepresenteerd als blokken die in elkaar kunnen worden geklikt.
- Intuïtieve interface: Kinderen kunnen programma's maken door blokken te slepen en neer te zetten, zonder code te hoeven typen.
- Onmiddellijke feedback: Veranderingen in de blokken leiden direct tot zichtbare resultaten, wat helpt bij het begrijpen van oorzaak en gevolg.

Gebruik
- Voorbeelden: Scratch, Blockly
- Toepassingen: Blokgebaseerde talen worden vooral gebruikt in educatieve contexten om kinderen de basisprincipes van programmeren te leren.

Geschiktheid voor kinderen
Blokgebaseerde talen zijn zeer geschikt voor jonge kinderen en beginners. Ze bieden een speelse en intuïtieve manier om te leren programmeren zonder de complexiteit van syntaxisfouten.

Code Voorbeeld (Scratch)
```scratch
when green flag clicked
repeat 10
    move 10 steps
    if on edge, bounce
```

Conclusie
Elk programmeerparadigma heeft zijn eigen sterke en zwakke punten, afhankelijk van de context waarin het wordt gebruikt. Voor het onderwijzen van kinderen zijn visuele en eenvoudigere programmeertalen vaak het meest effectief, zoals een blok gebaseerde, visuele programmeertaal die elementen van het imperatieve en objectgeoriënteerde paradigma combineert. Naarmate kinderen meer ervaring opdoen, kunnen ze worden geïntroduceerd in meer complexe paradigma's zoals objectgeoriënteerd en functioneel programmeren, afhankelijk van hun interesses en vaardigheden.