Informaticadocent & Multi Channel Learning

NIOC proceedings

3 en 4 november 2004
Groningen

Rein Smedinga
Jos Tolboom
editors

Ontmoetingsplaats voor informaticaonderwijs
www.nioc.nl

NIOC 2004 proceedings

Rein Smedinga
Jos Tolboom
editors

3 en 4 november 2004
Groningen

ISBN 90 5452 124 4
Uitgeverij Passage, Groningen
Omslag: Flip Drukker, Peize

Proceedings mede mogelijk gemaakt door:

UNIVERSITEIT VAN AMSTERDAM

Hogeschool van Amsterdam

Inhoudsopgave

1. Inleiding

Voor u liggen de proceedings van NIOC2004, gehouden op 3 en 4 november 2004 in Groningen als zevende in een rij van congressen over informaticaonderwijs. Het eerste NIOC vond plaats in Maastricht in 1990. Daarna in 1992 opnieuw in Maastricht, in 1994 en 1997 in Den Haag en in 1999 en 2002 in Enschede.

Moesten de voorgaande NIOC's het nog doen met congresbundels waarin samenvattingen van de presentaties waren opgenomen, NIOC2004 is het eerste congres met "echte" proceedings, bestaande uit (wetenschappelijke) artikelen geschreven naar aanleiding van de gepresenteerde bijdragen.

Naast deze proceedings is er een speciaal nummer van het tijdschrift TINFON verschenen, gewijd aan NIOC2004 en een aantal artikelen over bijdragen aan NIOC2004 bevattend. Daarnaast heeft het tijdschrift Informatie het nummer van oktober 2004 deels gewijd aan NIOC2004.

Het NIOC-bestuur bedankt alle mensen die hebben bijgedragen aan NIOC2004 en in het bijzonder de auteurs van de bijdragen in deze proceedings. Bijzondere dank is er voor de stichting Edict en voor de Universiteit van Amsterdam- Hogeschool van Amsterdam voor het willen subsidiëren van deze uitgave.

Het bestuur en de leden van de programmacommissies van NIOC2004 wensen u veel leesplezier toe.

2. Informaticadocent en multichannel learning

NIOC2004 in Groningen is de zevende in een rij van congressen over informaticaonderwijs. Tijdens de congressen in Maastricht (1990 en 1992), Den Haag (1994 en 1997) en Enschede (1999 en 2002) nam het aantal bezoekers langzaam af en het onderwerp van de congressen langzaam te veranderen van *onderwijs in de ict* naar *ict in het onderwijs*.

Terug naar de kern

Met NIOC2004 wilden we weer terug naar de kern, het informatica (of ict-)onderwijs. Daarbij willen we bijdragen over bijvoorbeeld e-learning niet bij voorbaat uitsluiten, zolang dit niet het hoofdthema is. Het eerste deel van het uiteindelijke thema "informaticadocent en multichannel learning" geeft aan dat de informaticadocent centraal moest staan in NIOC2004. Daarnaast wilden we een congres waarbij die docent zowel zou kunnen halen als brengen, waarbij we andere werkvormen dan alleen presentaties zouden willen stimuleren en waarbij we het congres vooral ook als ontmoetingsplaats van informaticadocenten zouden willen zien. Dit komt in ons thema naar voren in het tweede deel: "multichannel learning".

Het bestuur was zich er vanaf het begin van bewust dat NIOC2004 slechts mogelijk zou zijn als we de achterban bij onze ideeën zouden kunnen betrekken. Hiertoe is geprobeerd een groot aantal instellingen bij NIOC2004 te betrekken. We hebben onze doelgroepen opgedeeld in zogenaamde ontmoetingsplaatsen: WO-onderwijs, HBO-onderwijs, MBO/VBE-onderwijs, VO-onderwijs en onderwijs bij de bedrijfsopleiders. In elk van deze ontmoetingsplaatsen hebben we geprobeerd samenwerkingspartners te vinden die met ons wilden meedenken over de inrichting van NIOC2004 en, minstens zo belangrijk, ook het organiseren van evenementen tussen de NIOC's. Deze continuïteit noemen we: NIOC tussen de NIOC's.

Uiteindelijk hebben we vijfentwintig samenwerkingspartners gevonden, die elders in deze proceedings staan vermeld, en hebben we gezamelijk met deze partners onder meer het eerder genoemde thema bedacht en gewerkt aan invulling van congresonderdelen.

Bewust hebben we per ontmoetingsplaats een programmacommissie opgericht, om zo in elk gebied een aantal deskundigen bijeen te brengen die hun achterban zouden kunnen beïnvloeden en bijdragen voor NIOC2004 zouden kunnen vinden.

De samenwerkingspartners en de verschillende programmacommissies hebben een sneeuwbaleffect van betrokkenen bewerkstelligd. Een sneeuwbal die oorspronkelijk begon met eerste kennismakende gesprekken tussen onder anderen Hans Frederik, Trudy Berends, Jos Tolboom en Rein Smedinga. De sneeuwbal werd vervolgens groter doordat het gehele NIOC-bestuur zich achter de ideeën schaarde, en nog groter door de samenwerkingspartners en programmacommissies. Het sneeuwbaleffect was compleet toen deze laatsten hun achterban enthousiast maakten voor het NIOC volgens deze plannen.

Meer dan een congres

NIOC wil meer zijn dan eens in de twee jaar een congres over informaticaonderwijs. We willen, op het gebied van informaticaonderwijs kennis uitwisselen, ook buiten het congres om. Daarvoor is de website (www.nioc.nl) opgericht met daarbinnen een digitaal forum en proberen we, met behulp van de samenwerkingspartners, te komen tot een discussieplatform middels fysieke ontmoetingen door het hele land.

De samenwerkingspartners hebben elkaar in 2004 inmiddels ontmoet in Utrecht, Arnhem en Rotterdam en meer ontmoetingen staan op het programma.

NIOC speelt daarnaast een actieve rol in de oprichting van het IOIO (Instituut voor Ontwikkeling van Informaticaonderwijs), op moment van schrijven nog in ontwikkeling, dus feitelijk IOIOio, met als doel om de kwaliteit en de toegankelijkheid van het informaticaonderwijs op alle niveaus te bevorderen.

NIOC speelt eveneens een rol in het surf-project Connecticut (met de Rijksuniversiteit Groningen en de Hanzehogeschool Groningen), waarin onderzoek wordt gedaan naar competentiegericht curriculumontwerp.

CAIO: contextafhankelijk ict-onderwijs

Henk Plessius, Frans Vodegel en Sander Muizelaar,
Hogeschool van Utrecht

Dit artikel gaat in op het project CAIO: contextafhankelijk ict-onderwijs en is eerder verschenen in OnderwijsInnovatie nr. 3/2004, p. 33/36-37. Het project CAIO is een van de pijlers van het sectorprogramma informatica onderwijs in Nederland.

Het ict-onderwijs in het hbo en wo is in beweging; er vindt een omschakeling plaats van een groot aantal bverschillende opleidingen naar een brede bachelor of ict (Bict). Zo'n brede opleiding kenmerkt zich door een gemeenschappelijk kerncurriculum, aangevuld met een aantal uitstroomprofielen. Kenmerkend voor deze profielen is dat de ict in de context van het beroep wordt toegepast. Dat vraagt om een andere opzet van het ict-onderwijs. Het CAIO-project wil hierop inspelen.

De bama-structuur in het hoger onderwijs leidt ertoe dat curricula herzien worden. Zo ook het curriculum van de ict-opleidingen. Daarnaast is er een groeiende behoefte om het beroepsonderwijs competen-tiegericht op te bouwen, omdat alleen zo de relatie tussen onderwijs en beroepenveld kan worden versterkt [van Asselt, 2004]. Voegen we daar aan toe nieuwe didactische inzichten en de komst van nieuwe elektronische hulpmiddelen, dan zien we onderwijs ontstaan waarbij studenten, al dan niet in multidisciplinair samengestelde groepen, werken aan authentieke opdrachten. De daarvoor benodigde kennis is 'op afroep' beschikbaar. Dit meer vraaggestuurde onderwijs leidt ertoe dat studenten niet meer allemaal op hetzelfde ogenblik dezelfde ervaring opdoen: er ontstaan individuele leerroutes. Deze leerroutes bewegen zich in de praktijk naar toepassingsgebieden van de ict, zoals: techniek, communicatie, bedrijfskunde of gezondheidszorg. In dit artikel zullen we verder spreken van een 'context': een beroepspraktijk waarin de ict'er aan het werk is.

Specialisatie

Voor ict-opleidingen betekent deze ontwikkeling dat het curriculum zal gaan bestaan uit een voor iedereen gelijke, contextonafhankelijke kern, aangevuld met een specialisatie in een context waarbij individuele accenten kunnen worden gelegd. In dit verband is ook de ontwikkeling in het hbo naar een brede bachelor of ict (Bict) te plaatsen waarbij gedacht wordt aan een verdeling van 40 procent algemene ict-vakcompetenties, 30 procent toe-spitsing op de context en nog eens 30 procent in combinatie met aanpalende werkvelden [Vissers, 2004]. CAIO, Context Afhankelijk Ict-Onderwijs, speelt op deze ontwikkelingen in. Dit project vormt een van de belangrijkste pijlers van het sectorprogramma infor-maticaonderwijs Nederland (SPIoN) [Vodegel, 2004] dat inmiddels aan de Digitale Universiteit (DU) aangeboden is.

Een voorbeeld van contextafhankelijkheid: het begrip 'requirements' zou als volgt uitgelegd kunnen worden:

- voor de student Bedrijfskundige Informatica:

De requirements beschrijven in natuurlijke taal, zo nodig aangevuld met diagrammen, welke functies het systeem moet kunnen uitvoeren.
• voor de student Technische Informatica: De requirements zijn een exacte specificatie van de functies, services en operationele constraints van het systeem.

De doelstelling in het CAIO-project is een contentdatabase te ontwikkelen die via elektronische leeromgevingen 'benaderd' kan worden. In deze database zal te zijner tijd een grote hoeveelheid relevante gemeenschappelijke ict-kennis van de brede Bict-opleiding beschikbaar zijn, waarbij de kennis vanuit verschillende contexten aangesproken kan worden.

CAIO en didactiek

Een belangrijk uitgangspunt van het CAIO-project is dat er didactisch neutraal ontwikkeld wordt. Zoals hierboven aangegeven werd, is CAIO in eerste instantie ontstaan vanuit de behoefte om beter in te kunnen spelen op de eisen die competentiegericht onderwijs aan de content stelt, maar wil het ook bruikbaar kunnen zijn in meer aanbodgericht en docentgecentreerd onderwijs. In competentiegericht onderwijs waarin veel wordt gewerkt met authentieke beroepsopdrachten, moet de CAIO-database voor studenten dè plek zijn waar zij kwalitatief hoogwaardige informatie kunnen vinden. Deze informatie zullen ze bijvoorbeeld willen raadplegen als ze bij de uitvoering van de opdracht op problemen stuiten waar ze niet direct een oplossing voor hebben. In dat soort gevallen moet CAIO uitkomst bieden. Dit zal uiteraard vaak geen pasklaar antwoord zijn op het probleem, maar de studenten wel een oplossingsrichting aangeven. Essentieel hierbij is dat de CAIO-database gemakkelijk en via vele ingangen doorzoekbaar is, zodat de studenten de gewenste informatie gemakkelijk vinden kunnen. CAIO faciliteert in competentiegericht onderwijs dus met name het 'just-in-time'-leren. In meer aanbodgericht onderwijs fungeert CAIO veel meer als een flexibel samen te stellen tekst- en werkboek. Zowel studenten als docenten kunnen zelf modules samenstellen op basis van de leereenheden (learning objects) die in CAIO zijn opgenomen. Zo´n module noemen we een educational component (EC): een op zichzelf staande eenheid van studie (gezien vanuit het oogpunt van de student) die op zichzelf bestudeerd (en eventueel getoetst) kan worden [de Bruin et al, 2004]. Zo'n EC zal over het algemeen een omvang van 3 à 4 studie-uren hebben, overeenkomend met één dagdeel studie (of de hoeveelheid stof die een ervaren docent in een blokuur aanbiedt).
Een EC heeft niet alleen een inhoudelijk, maar ook een contextaspect. Een EC moet betekenis hebben voor de student. In deze situatie biedt CAIO, in vergelijking met een papieren tekst- en werkboek, naast flexibiliteit, met name voordelen op het gebied van beheer en tijd- en plaatsonafhankelijke benaderbaarheid.

Architectuur

In de architectuur van CAIO worden kleinere elementen in hun onderlinge samenhang onderscheiden. Om te beginnen worden twee dimensies onderscheiden: de inhoudelijke dimensie, waarin de onderwerpen (ict-gebieden) waar CAIO uiteindelijk in moet voorzien, een rol spelen. Vooralsnog zijn hier de volgende vier ict-gebieden onderscheiden: databases, programmeren, computer-techniek & netwerken en systeem-

ontwikkeling. Daarnaast is er de contextuele dimensie, waarin de verschillende contexten een plaats krijgen. In aansluiting op het bestaande ict-onderwijs (zie de tekstbox hieronder met voorbeelden van opleidingen) zullen de volgende contexten allereerst ontwikkeld worden: bedrijfskundig, informatiekundig (applicatiebouw) communicatief en technisch.
Al met al is zo een indeling van het totale CAIO-project gerealiseerd in vier inhoudsgebieden en vier contexten, die vervolgens worden opgeknipt in learning objects.

Voorbeelden van ict-opleidingen:

- **Bedrijfskundig**: Bedrijfskundige Informatica, Informatiemanagement, ...
- **Informatiekundig**: Informatica, Information Engineering, ...
- **Communicatie en media**: Communicatiesystemen, Mediatechnologie, Interactieve Media, ...
- **Technisch**: Technische Informatica, Computertechniek, ...

Voorbeelden van ict-opleidingen:

- **Bedrijfskundig**: Bedrijfskundige Informatica, Informatiemanagement, ...
- **Informatiekundig**: Informatica, Information Engineering, ...
- **Communicatie en media**: Communicatiesystemen, Mediatechnologie, Interactieve Media, ...
- **Technisch**: Technische Informatica, Computertechniek, ...

Ontwikkelaanpak
De ontwikkeling van CAIO is deze maand begonnen. De lastige stap die de ontwikkelteams moeten maken, is om te komen tot consensus over het contextonafhankelijke gedeelte van een onderwerp, en daarmee ook tot consensus over de contextafhankelijke aspecten. Dit vergt een diepgaande analyse van het onderwerp vanuit de verschillende contexten: waar zit de overlap, en waar zitten de verschillen?
Een onderwerp wordt tijdens dit proces opgeknipt in twee typen leerobjecten: contextonafhankelijke leerobjecten, zogenaamde COLO's, waarin zaken worden behandeld die voor alle contexten relevant zijn (al kunnen zer indien gewenst onderdelen aan worden toegevoegd voor specifieke doelgroepen zoals het voorbeeld in de textbox illustreert), en contextafhankelijke leerobjecten, CALO's, waarin zaken behandeld worden die alleen voor één (of meerdere) context(en) van belang zijn.

Voor het eerder aangehaalde voorbeeld van de requirements zou de COLO er als volgt uit kunnen zien:

De requirements zijn een [B: nauwkeurige beschrijving] [T: exacte specificatie] van de functies [T: services en constraints] van het systeem.
In CALO´s kunnen dan voorbeelden voor de verschillende doelgroepen gegeven worden.

Naast de overeenstemming over de contextonafhankelijke onderdelen moeten er ook besluiten worden genomen over de mate van diepgang waarmee deze componenten behandeld zullen worden. Immers, in veel gevallen zal een opleiding als Informatica of Technische informatica dieper op ict-aspecten ingaan dan bijvoorbeeld Bedrijfskundige informatica. Die verdiepingsslagen zijn contextafhankelijk en worden dus in

contextafhankelijke leerobjecten behandeld. De ontwikkelaars moeten hierbij hun aandacht primair kunnen richten op de inhoud. De technische realisatie ligt nadrukkelijk bij het project. Hier wordt gezorgd voor de inrichting van een efficiënte contentontwikkelstraat, die ook voor andere kennisdomeinen en sectoren kan worden ingezet. Dit geldt dan ook als één van de zogenaamde deliverables van het project. Het ontwikkelproces levert een database op van COLO's en CALO's die met behulp van metadata gelabeld is en daarmee gemakkelijk doorzoekbaar. Zoals gezegd zal in competentiegericht onderwijs deze database vooral individueel door de student doorzocht worden op het niveau van learning objects, terwijl in meer aanbodgericht onderwijs het mogelijk is dat docent of student met deze learning objects zogenaamde educational components samenstelt.

Het CAIO-project wordt uitgevoerd door de Hogeschool van Utrecht, in nauwe samenwerking met Saxion Hogescholen, de Universiteit Twente, de Hogeschool van Rotterdam en de Fontys Hogescholen. Het ligt in de bedoeling vanuit de ontwikkelgroepen koppelingen te maken naar de kenniscommunities van het hbo-I. Zo kan de community 'programmeren' bijvoorbeeld een rol spelen als reviewgroep voor de CAIO-ontwikkelgroep op dit onderwerp. Behalve dat daardoor het contact tussen ict-docenten in ons land wordt gestimuleerd, wordt zo ook het draagvlak voor de CAIO-onderwijsproducten groter.

Natuurlijk is het zo dat 'the proof of the pudding is in the eating' is, maar we zijn als projectvoorbereiders erg enthousiast over dit project. Met name omdat dit project beoogt:

- in hoofdlijnen overeenstemming te bereiken over ict-content tussen verschillende opleidingen,
- ict-kennis in het perspectief van het toekomstige beroep te plaatsen,
- een integrale contentdatabase te vullen waarin een behoorlijke hoeveelheid kennis via een elektronische leeromgeving toegankelijk wordt. Deze database kan gebruikt worden als lesmateriaal in verschillende didactische settings, maar ook als naslagwerk,
- gebruik en beheer van content te verenigen in één model (docenten en studenten kunnen zelf ontwikkelde content op laten nemen in de database), en
- een ontwikkelstraat op te zetten die als voorbeeld voor vergelijkbare projecten kan gelden.

We hopen met deze korte schets meer mensen enthousiast te krijgen voor onze plannen die kunnen bijdragen aan een transformatie van het ict-onderwijs.

Literatuur

- Asselt, R. van et al. (2004). *Doorstroming, van werkelijkheid naar droom*. Interne publicatie Saxion Hogeschool, Enschede
- A. Vissers et al. (2004). *Position paper Bachelor of Information and Communication*. Stichting HBO-I, Amsterdam
- F. Vodegel (2004). Programma Outline Sectorplan ICT-opleidingen in Nederland. Digitale Universiteit, Utrecht
- L. De Bruin, H. Plessius, P. Ravesteyn (2004). *e-Learning in higher Education: a Casestudy.* Paper presented at eLearning Results 2004, Sestri Levante

Websites

- Digitale Universiteit Nederland: www.digiuni.nl
- Het Picture-project van de Hogeschool van Utrecht: www.picture.hvu.nl
- Het onderwijskundig adviescentrum Cetis van de Hogeschool van Utrecht: www.cetis.hvu.nl

Note

Dit artikel is eerder gepubliceerd in OnderwijsInnovatie 3/2004, p 33/36-37

SIM-PL, auteursomgeving voor digitale componenten

Ben Bruidegom en Wouter Koolen-Wijkstra, AMSTEL-insituut UvA

Samenvatting
SIM-PL is een auteursomgeving om componenten te construeren en te simuleren voor cursussen zoals Digitale techniek, Computerarchitectuur en Computerorganisatie. SIM-PL is te gebruiken in het hele spectrum van abstracties dat bij deze cursussen aan bod komt dus van poortschakelingen tot pipeline processoren. Er wordt een bibliotheek van herbruikbare voorbeeldcomponenten en schakelingen meegeleverd.
Keywords: Digitale techniek, Computerarchitectuur, Computerorganisatie

Inleiding

SIM-PL is een onderwijsgericht modellerings- en simulatiepakket voor digitale componenten. Het wordt met succes toegepast in het informatica en kunstmatige intelligentie curriculum aan de Universiteit van Amsterdam. Dit artikel is als volgt opgebouwd: Eerst leggen we uit wat digitale componenten zijn, en hoe deze componenten in de computer gerepresenteerd worden. Daarna gaan we in op de specifieke eigenschappen van onze simulator. Vervolgens komt onze toekomstvisie m.b.t. SIM-PL aan bod, en als laatste beschouwen we de sterke punten van SIM-PL, en de daar op natuurlijke wijze uit voortvloeiende toepassingen.

Digitale Componenten

Digitale componenten zijn hardwareschakelingen waarvan alle in/uitgangen slechts de waarden 0 of 1 kunnen aannemen. De waarde van de uitgangen is een functie van de waarden van de ingangen in de tijd. Hierbij speelt de *propagation delay* (het tijdsverschil tussen een verandering in de inputwaarde tot de bijbehorende verandering in de outputwaarde) een belangrijke rol. Digitale componenten worden *hiërarchisch* opgebouwd. Bijv. poort → flipflop → register → registerfile van een processor. SIM-PL geeft inzicht in de werking van digitale componenten door het waardeverloop op ingangen, uitgangen en interne verbindingen exact in de tijd te simuleren.

Componenten[i] Construeren

In SIM-PL zijn er, parallel aan de hiërarchische opbouw van fysieke digitale componenten, twee typen componenten:

- Basiscomponenten. Deze atomaire componenten zijn de functionele bouwblokken van de schakeling
- Complexe componenten. Deze zijn opgebouwd uit subcomponenten en verbindingen

Constructie basiscomponent

De constructie van een *basiscomponent* bestaat uit:

- Tekenen basisfiguur
- Vastleggen plaats inputs en outputs
- Programmeren van de relatie tussen inputs en outputs (zie *Events*)
- Programmeren van een eventuele geheugenfunctie
- Instellen van de propagation delay

Events

Componenten worden geactiveerd door de volgende events:

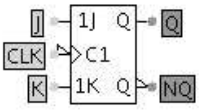
- INIT legt de begintoestand van de component vast
- INPUT-CHANGE reageert op een signaalverandering van één of meer *ingangen*.
- CLOCK-RISING reageert op een positieve klokflank
- CLOCK-FALLING reageert op een negatieve klokflank

Aan elk van deze events kan een programma worden verbonden. (Zie *Interne programmeertaal*) Dit programma wordt uitgevoerd als de bijbehorende event optreedt, en definieert op deze wijze de functionaliteit van de component.

Interne programmeertaal: nBit

Componenten worden in een taal geprogrammeerd met een syntax die sterk op C/C++/Java lijkt. Het basis datatype is een getal van *n* bits. Dit wordt gebruikt om een aantal parallelle signaallijnen te representeren.

Als voorbeeld is hieronder een JK–flipflop[ii] weergegeven met een deel van de bijbehorende door de editor gegenereerde xml-code.



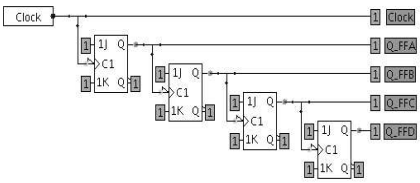

```
<MEMORY>
<STORAGE NAME="m" BITS="1" SIZE="1"/>
<STORAGE NAME="clock" BITS="1" SIZE="1"/>
</MEMORY>
<INTERNALS DELAY="2">
<ACTION EVENT="INPUT_CHANGE">
{
  if( !CLK && clock[0] ) {  // negative edge
  if( J && !K ) m[0]= 1;
  if( !J && K ) m[0]= 0;
  if( J && K ) m[0]= !m[0];
}
 clock[0] = CLK;
 Q= m[0];
 NQ=!m[0];
}
</ACTION>
```

Constructie complexe component

De constructie van een *complexe component* bestaat uit:

- Ophalen en configureren van de subcomponenten van de schakeling.
- Vastleggen plaats inputs en outputs van de gehele schakeling .
- Trekken van de verbindingen tussen de diverse inputs en outputs van de subcomponenten onderling en met de in- en outputs van de schakeling. Op een verbinding kan een 'waypoint/probe' worden geplaatst die de toestand van het signaal op dat punt tijdens de simulatie weergeeft.

Als voorbeeld is hieronder een asynchrone teller samengesteld uit JK-flipflops weergegeven.



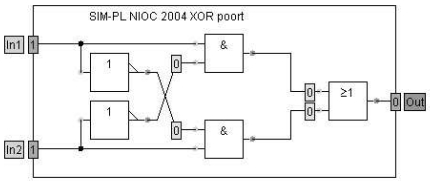
Componenten Simuleren

SIM-PL simuleert het exacte signaalverloop door een schakeling in de tijd. De gebruiker kan de logische waarden op iedere uitgang en verbinding bekijken en vrije ingangen aansturen. Voor het aanleveren van complexe signalen maakt SIM-PL gebruik van twee compilers[iii]:

- De universele compiler
- De generieke assembler compiler

Universele compiler

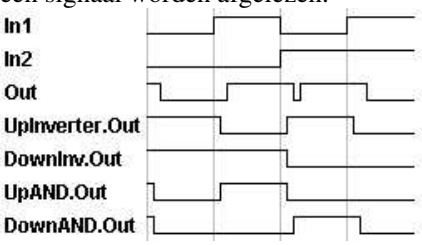
De universele compiler vertaalt laag niveau code in een multikanaal discreet signaal voor willekeurig welke gesimuleerde component. Dit is in het bijzonder handig voor het genereren van waarheidstabellen. Een voorbeeld hiervan is de onderstaande component, een exclusive or-poort.



De voor dit doel ingebouwde optie "generate truth table" produceert onderstaande code. [iv]

```
00: In1= 0;
00: In2= 0;
10: In1= 1;
10: In2= 0;
20: In1= 0;
20: In2= 1;
30: In1= 1;
30: In2= 1;
```

In de volgende figuur is het tijdvolgordediagram weergegeven van dit "programma". De signalen op de in- en uitgangen worden standaard in de tijd weergegeven. Door op een verbinding te klikken tussen twee componenten wordt ook het signaal van deze verbinding weergegeven. De propagatietijd van alle poorten is op 1 gesteld. De initiële waarde van alle ingangen en uitgangen is 1. Met de muis kan de waarde en het tijdstip van een signaal worden afgelezen.



Generieke assembler compiler

De *generieke assembler compiler* vertaalt – gegeven een componentspecifieke instructieset definitie - een assemblerprogramma voor de bijbehorende processorcomponent.

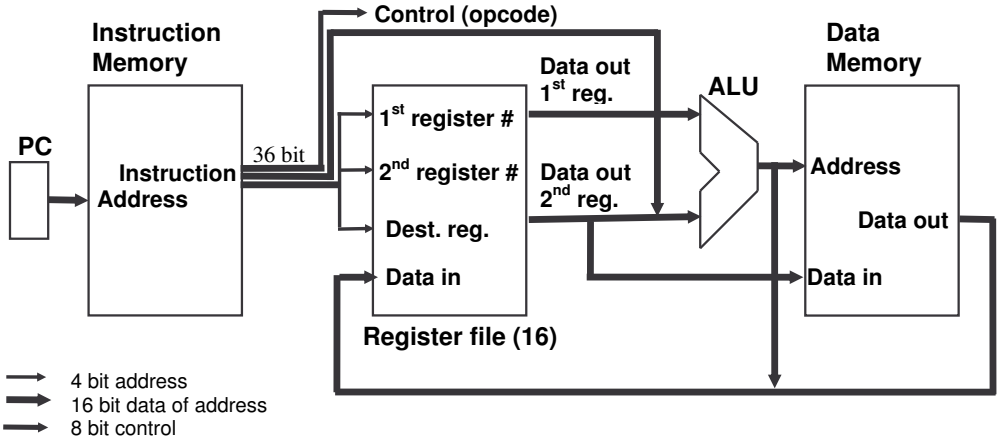
Als voorbeeld is op de volgende bladzijde een sterk vereenvoudigd model van een "Harvard machine[v]" (zie referentie) weergegeven. Deze architectuur bestaat uit vijf hoofdcomponenten: Program Counter (PC), Instruction Memory, Register file, ALU en Data Memory. Iedere instructie wordt in één clockcycle uitgevoerd. Eronder ziet u de realisatie van dit model in SIM-PL. De simulator laat de status zien na het uitvoeren van de instructie LI $1, 0x01FD (Load Immediate register 1 met het getal $01FD_{HEX}$).

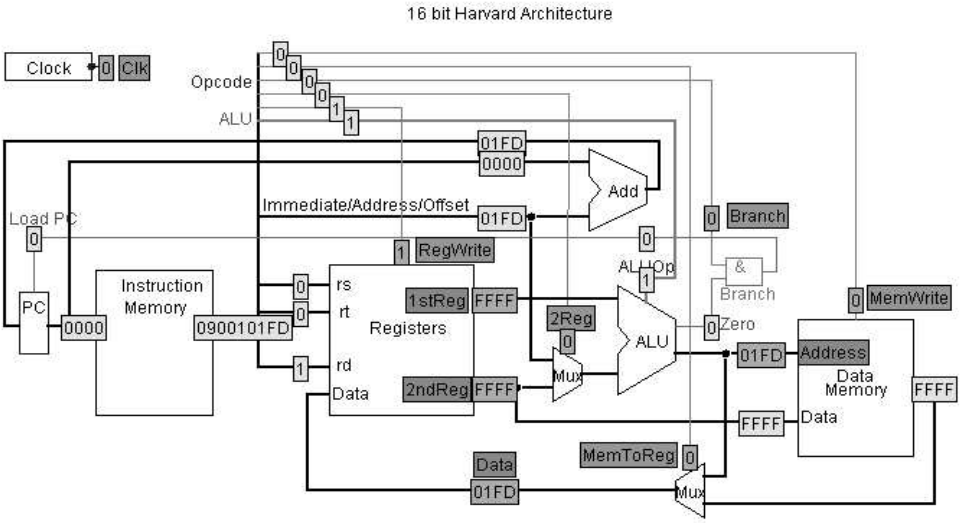
Door op één van de componenten te klikken wordt de status van deze component weergegeven. Zo kan men bijvoorbeeld de status van de Register file of het Data Memory bekijken tijdens het executeren van het programma.

Op de volgende pagina ziet u een voorbeeldsectie uit de instructiesetdefinitie, en daaronder een simpel programma voor deze architectuur. Het programma slaat 5 getallen op in het datageheugen

en telt deze daarna op. De universele compiler wordt aangeroepen op dit programma. De eerste regel vertelt hem waar het instructiesetdefinitiebestand te vinden is. De inhoud van dit bestand bepaalt hoe de rest van het programma wordt geïnterpreteerd. Door het aanpassen van de instructiesetdefinitie kan hetzelfde programma op meerdere processorarchitecturen worden gebruikt.

Er zijn meerdere processormodellen geïmplementeerd, de ingewikkeldste is een MIPS processor met 5 pipeline stages. Deze wordt gebruikt in het lesprogramma voor de informaticastudenten.





```
# bestand: "16bitHarvard.asm.txt"

# unary arithmetical operators
#format   ARITH1  OP:0d5, ALU:0d3, RS:0d4, RT:0d4, RD:0d4, IMM:0d16 | rd:0d4, rt:0d4
#def NOT  ARITH1  " OP = 0b00011; ALU = 0x0; RS = 0; RT = rt; RD = rd; IMM = 0;"
#def MOVE ARITH1  " OP = 0b00011; ALU = 0x1; RS = 0; RT = rt; RD = rd; IMM = 0;"

# binary arithmetical operators
#format   ARITH2  OP:0d5, ALU:0d3, RS:0d4, RT:0d4, RD:0d4, IMM:0d16 | rd:0d4, rs:0d4,
rt:0d4
#def SUB  ARITH2  " OP = 0b00011; ALU = 0x2; RS = rs; RT = rt; RD = rd; IMM = 0;"
#def ADD  ARITH2  " OP = 0b00011; ALU = 0x3; RS = rs; RT = rt; RD = rd; IMM = 0;"
#def XOR  ARITH2  " OP = 0b00011; ALU = 0x4; RS = rs; RT = rt; RD = rd; IMM = 0;"
#def OR   ARITH2  " OP = 0b00011; ALU = 0x5; RS = rs; RT = rt; RD = rd; IMM = 0;"
#def AND  ARITH2  " OP = 0b00011; ALU = 0x6; RS = rs; RT = rt; RD = rd; IMM = 0;"

# transfer data to and from data memory to registers
#format    LOADSTORE  OP:0d5,  ALU:0d3,  RS:0d4,  RT:0d4,  RD:0d4,  OFFS:0d16|  rd:0d4,
offs:0d16, rs:0d4
#def LW  LOADSTORE "OP=0b01001; ALU=0x3; RS = rs; RT = 0; RD = rd; OFFS = offs;"
#def SW  LOADSTORE "OP=0b10000; ALU=0x3; RS = rs; RT = rd; RD = 0; OFFS = offs;"
```

```
#include "16bitHarvard.asm.txt"

# Add 5 values stored in memory locations
5..9 and store the result in
# the first free location 10

        LI $5, 0x5  # 5 values
        LI $6, 0    # Start at element 0
        LI $0, 0    # Clear $0

# Store 0d10 in address 0d5, 0d11 in
address 0d6, etc.
        LI $4, 0d10
        SW $4, 0d5($0)
        LI $4, 0d11
        SW $4, 0d6($0)
        LI $4, 0d12
        SW $4, 0d7($0)
        LI $4, 0d13
        SW $4, 0d8($0)
        LI $4, 0d14
        SW $4, 0d9($0)

loop:   LW $1, 0d5($6)
        ADD $0, $0, $1
        ADDI $6, $6, 1
        BEQ $6, $5, end
        BRA loop

end: SW $0, 0d5($6)
```

Verbeteringen, uitbreidingen

SIM-PL is nog in ontwikkeling. De volgende wensen zijn geuit:

- Gebruiksvriendelijker Editor (bijv. undo-optie).
- Implementatie om Micro-programmeren mogelijk te maken.
- Toevoegen C-compiler om aansluiting te maken bij het vak “Operating Systems”.
- Toevoegen van Componenten en Architecturen uit de meest gebruikte boeken.
- Geschikt maken voor het middelbaar onderwijs.

Visie om dit te realiseren

Vooral vanuit het HBO maar ook uit het VO is interesse getoond om SIM-PL te gaan gebruiken. De auteurs zoeken dan ook partners om bijv. in Digitale Universiteit-verband SIM-PL verder te realiseren.

Onderwijsrelevantie

De krachtige ontwerpmethodiek van SIM-PL sluit naadloos aan bij het conceptueel denken over schakelingen van verschillende abstractieniveaus. SIM-PL maakt het mogelijk exact het gewenste detail te representeren en te simuleren. Het softwarepakket simuleert schakelingen van een paar poorten tot complete processoren met geheugen-hiërarchieën.

Als voorbeeldtoepassing werd een 16 bit Harvard model gepresenteerd. Studenten kunnen hiervoor assembler-code schrijven, de code assembleren en executeren voor dit door de docent geconstrueerde computermodel met instructieset. Dit tot zijn essenties gereduceerde model van een computer is o.a. uitgeprobeerd met VWO-leerlingen. In korte tijd kon een groot deel van deze leerlingen met het model omgaan en werd de werking ervan begrepen.

Andere redenen zijn:

- SIM-PL is een auteursomgeving geschikt voor docenten, studenten en scholieren.
- SIM_PL dicht het "gat" tussen de vakken "Digitale techniek" waarin poorten, flipflops, tellers etc. worden behandeld en "Computerarchitectuur en computerorganisatie" waarin complete (pipeline)processoren aan de orde komen.
- SIM-PL is beschikbaar onder GPL licentie (Free Software) en is te downloaden via www.science.uva.nl/~benb/SIM-PL

Referentie

Boek: David A. Patterson, John L. Hennessy, Computer Organisation And Design Chapter 5.4, Elsevier

[i] Vanaf dit moment wordt met 'component' de representatie in de simulatie bedoeld, dit i.t.t. het fysieke circuit.

[ii] Deze component kan worden opgebouwd uit subcomponenten. Om te laten zien dat het insteekniveau van basiscomponenten volledig variabel is, is dat hier niet gedaan.

[iii] Met 'compiler' bedoelen we een sofware component die instructies in een specifieke taal omzet in signalen voor de op dat moment gesimuleerde component. Het abstractieniveau van de taal hangt natuurlijk af van de component.

[iv] Links staat het tijdstip waarop de ingangen hun signalen krijgen aangeboden.

[v] Een Harvard machine heeft twee geheugens: één voor instructies en één voor data.

Computer Vision Lab NHL

Jaap van de Loosdrecht, Noordelijke Hogeschool Leeuwarden

Samenvatting
Het Computer Vision Lab Van de Noordelijke Hogeschool Leeuwarden is in 1994 gestart en is inmiddels uitgegroeid tot een goed uitgerust laboratorium met een grote verscheidenheid aan camera's, lenzen, belichtingsapparatuur, framegrabbers en beeldverwerkingsoftware. Het Computer Vision Lab heeft sinds zijn start bij bedrijven meer dan vijfentwintig projecten op het gebied van Computer Vision met succes uitgevoerd. Verder is er een onderwijsmodule ontwikkeld op het gebied van Computer Vision die zowel aan reguliere studenten als aan cursisten uit het bedrijfsleven wordt aangeboden.
Keywords: Computer Vision, curriculum ontwikkeling, kennistransfer bedrijfsleven

Computer Vision
Computer Vision (ook wel beeldverwerking genoemd) houdt in dat met behulp van een computer beelden worden geïnterpreteerd die met een sensor (bijvoorbeeld een camera) zijn vastgelegd. De op deze wijze verkregen informatie kan vervolgens worden gebruikt om andere processen aan te sturen.
Voorbeelden daarvan zijn:

- kwaliteitscontrole
- positie- en oriëntatiebepaling
- sorteren van producten op lopende banden.

Voor veel bedrijven die producten maken of verwerken is visuele inspectie of meting belangrijk. Met behulp van Computer Vision is het in een groot aantal gevallen mogelijk om deze inspecties of metingen geautomatiseerd te laten uitvoeren. Dit zal in veel gevallen kunnen bijdragen aan een goedkoper, flexibeler en/of arbeidsvriendelijker productieproces.
In het verleden waren de kosten van computers die beschikten over de benodigde rekencapaciteit een grote belemmering voor het invoeren van Computer Vision systemen. Het afgelopen decennium hebben computers echter een geweldige toename in rekenkracht gekregen. Tegenwoordig heeft een standaard PC voor veel toepassingen vaak al voldoende rekenkracht om het probleem op te lossen.

Start Computer Vision Lab
Doelstellingen van het Computer Vision Lab bij de oprichting in 1994 waren het vergaren van kennis en expertise op het gebied van Computer Vision, het opbouwen van een kennisnetwerk en het opzetten en verzorgen van het nieuwe vak Computer Vision voor de studenten van de opleiding Informatica.
Het uiteindelijke doel van het Computer Vision Lab was om te komen tot een kennistransfer richting het bedrijfsleven door middel van het verrichten van betaald onderzoek en het uitvoeren van werkzaamheden in opdracht van dit bedrijfsleven.
De start was bescheiden; lezingen werden gegeven, de eerste camera's werden aangeschaft en er werd een begin gemaakt met kleine onderwijsprojecten.
Al gauw diende het eerste commerciële project zich aan. Daardoor ontstond ruimte voor verdere investeringen en konden de medewerkers van het lab hun kennis verder ontwikkelen. Ook kreeg het vak Computer Vision binnen de opleiding Informatica gestalte. Door de hier verworven kennis op het gebied van Computer Vision beschikten de studenten over een hoger startniveau waardoor meer en meer complexe projecten konden worden uitgevoerd.

In de vorm van vierde jaarsprojecten en stage- en afstudeeropdrachten werden de studenten in toenemende mate bij het lab en zijn werkzaamheden betrokken.

Organisatie

Het Computer Vision Lab is een onderdeel van de Technologische Werkplaats van de NHL. Binnen deze Technologische Werkplaats worden multidisciplinair onderzoek verricht en werkzaamheden uitgevoerd voor het bedrijfsleven, dóór docenten en studenten van de opleidingen Informatica, Elektrotechniek en Werktuigbouwkunde.

Het Computer Vision Lab is een onderdeel van de afdeling Engineering van de NHL. De staf van dit lab bestaat uit een coördinator, een docent Informatica, een docent Werktuigbouwkunde en twee project-ingenieurs. Deze projectingenieurs zijn studenten van de opleiding Informatica die bij het lab afgestudeerd zijn op het gebied van Computer Vision en daar vervolgens in dienst zijn getreden. Verder werkt er een wisselend aantal stagiairs en afstudeerders in het Lab.

Veel van de uitgevoerde Computer Vision projecten hebben een multidisciplinair karakter.

Projecten en projectacquisitie

De projecten die het Computer Vision Lab uitvoert voor het bedrijfsleven zijn "toegepast onderzoek" projecten. Het lab is sterk in het uitvoeren van haalbaarheidsonderzoeken en het bouwen van prototypen. Aangezien een Hogeschool een onderwijsinstelling is valt het buiten haar werkzaamheden om bijvoorbeeld een prototype in productie te nemen en iets dergelijks als een 24-uurs storingdienst te onderhouden voor een product. Dit betekent dat er na de bouw van een prototype twee mogelijkheden zijn: de kennis wordt overgedragen aan de opdrachtgever, zodat hij zelf het apparaat in productie kan nemen en onderhouden, of er wordt gezamenlijk met de opdrachtgever naar een systeemintegrator gezocht die dit kan doen voor de opdrachtgever.

Voor het MKB in Noord Nederland kan vaak bemiddeld worden met betrekking tot het verkrijgen van subsidie.

Het Computer Vision Lab heeft inmiddels ruim vijfentwintig projecten succesvol kunnen afsluiten. Voorbeelden van deze projecten of sectoren waarin projecten zijn uitgevoerd:

- folie inspectie
- scheerapparaten inspectie
- toegangscontrole personen
- bloembolsorteermachine
- aardappelsorteermachine
- kalibratie van boorinrichting
- controle krassen op glasplaten
- controle pijpen van ultracentrifuge m.b.v. een neuraal net
- microbiologie
- zuivelindustrie
- intensieve veehouderij
- vleesverwerkende industrie

Klanten zijn zowel eenmanszaken als multinationals afkomstig uit Friesland en de rest van Nederland.

Onderwijs in Computer Vision

Het vak Computer Vision is de afgelopen jaren gegeven aan studenten van de opleidingen Informatica, Elektrotechniek en Werktuigbouwkunde aan de NHL, en aan cursisten uit het bedrijfsleven. De lesstof is in de loop der jaren geïnspireerd door ervaringen die zijn opgedaan bij de in opdracht van het bedrijfsleven uitgevoerde projecten. De speciaal voor Computer Vision ingerichte practicumzaal met daarin tien werkplekken met pc's, camera's en belichtingsapparatuur is uniek in de HBO wereld in Nederland.

De doelstelling van de cursus is dat de cursist kennis maakt en oefent met de volgende technieken en aspecten en deze kan toepassen bij het ontwerpen en bouwen van Computer

Vision toepassingen: beeldacquisitie, image math, geometric operators, contrast manipulation, segmentation, lineaire filters, edge detection, niet-lineaire filters, distance transform, Hough transform, 2D camera kalibratie, Fourier transform, kleurenbeeldverwerking, barcode-identificatie en classificatie met behulp van neurale netwerken.
De theorie wordt steeds afgewisseld met praktische oefeningen. Naast het oefenen met bewerkingen op beelden wordt er ook tijd besteed aan aspecten van beeldacquisitie, zoals keuze en instelling van camera's, framegrabbers, lenzen en belichting. Per twee cursisten is er een werkplek met pc, camera en belichtingsapparatuur.
De cursus wordt gegeven door docenten die werken bij het Computer Vision Lab. Tijdens de cursus wordt gebruik gemaakt van een beeldverwerkingspakket waarmee op comfortabele wijze geëxperimenteerd kan worden met de Computer Vision operatoren en waarmee scripts kunnen worden ontwikkeld en getest. Het cursusmateriaal is in het Engels geschreven. Dit cursusmateriaal is ook gebruikt voor een gastdocentschap aan de Universiteit Jaume I in Castello de la Plana in Spanje.
Via de website van het Computer Vision Lab [1] zijn de powerpoint presentaties, de bij de oefeningen gebruikte beelden en een demoversie van het gebruikte beeldverwerkingpakket beschikbaar voor persoonlijk gebruik [2].
In het reguliere onderwijs binnen de opleiding Informatica is het vak Computer Vision een 3e jaar keuzevak (vier studiepunten) en bestaat uit tien hoorcolleges van twee uur en tien dagdelen practicum.
Voor het bedrijfsleven wordt het vak Computer Vision als vijfdaagse cursus aangeboden. De eerstvolgende cursus wordt gegeven op 25 tot en met 29 april 2004. In deze cursus komt de theorie uit de reguliere cursus aan de orde met een aangepast practicum. De laatste middag werkt de cursist aan een integrerende opdracht. Daarbij wordt de cursist uitgenodigd om hiervoor zelf 'inspectie objecten' mee te nemen. Hij kan dan onder begeleiding werken aan een probleem uit zijn dagelijkse praktijk. Deze manier van werken wordt door de cursisten als zeer positief ervaren. Zie voor een brochure van deze cursus de aan het einde van dit artikel genoemde website [1].

Platform Beeldverwerking HBO

In samenwerking met de IOP Beeldverwerking (Senter Economische Zaken) heeft de NHL het Platform Beeld Verwerking HBO opgericht.
Dit platform heeft als doel:

- Een brugfunctie te vervullen voor het HBO bij het vergaren, uitwisselen en verankeren van de kennis en expertise op het gebied van beeldverwerking.
- Het als zodanig fungeren als kennis(sen)netwerk rond beeldverwerking in het HBO in Nederland.
- Een kennistransfer naar HBO instellingen en vervolgens naar het bedrijfsleven te bewerkstelligen
- Het stimuleren van contractactiviteiten op het gebied van beeldverwerking door HBO instellingen.

Geplande activiteiten van dit platform zijn:

- Het opleiden van HBO docenten tot docent Computer Vision. De docenten volgen de vijfdaagse cursus Computer Vision van de NHL. Het idee is dat de cursus Computer Vision van de NHL de basis gaat vormen voor het onderwijs bij de andere HBO instellingen.
- Het opzetten en bemannen van een helpdesk voor HBO instellingen die onderwijs geven op het gebied van beeldverwerking.
- Het verder ontwikkelen van een curriculum Computer Vision voor het HBO.

- Het opzetten en onderhouden van internetsite/mailing list.
- Het geven van cursussen beeldverwerking voor MKB.
- Het organiseren van themabijeenkomsten en workshops.

Om te stimuleren dat de kennis van beeldverwerking verspreid wordt naar andere hogescholen heeft de IOP Beeldverwerking besloten dat docenten van hogescholen een bijdrage van 50% kunnen krijgen op de kosten van de cursus Computer Vision van de NHL. Voor meer informatie hierover kunt u terecht bij IOP Beeldverwerking [3].

Besluit

Het Computer Vision Lab van de Noordelijke Hogeschool Leeuwarden heeft zich in de afgelopen tien jaar ontwikkeld tot een modern en goed uitgerust Laboratorium. Het is gebleken dat binnen bedrijven voor Computer Vision steeds meer toepassingsmogelijkheden zijn. In samenwerking met de IOP Beeldverwerking (Senter, Economische Zaken) heeft de NHL het Platform Beeld Verwerking HBO opgericht om de kennis op het gebied van beeldverwerking te verankeren in de maatschappij.

---

[1] website Computer Vision Lab NHL: www.engineering.tech.nhl.nl/computervision
[2] website Cursus Computer Vision NHL:
www.engineering.tech.nhl.nl/engineering/personeel/loosdrec/vision_course/index.html
[3] dhr HenkJan Reus IOP Beeldverwerking, 070 373 5371, H.J.W.Reus@senter.nl

Peer Assessment: laat studenten elkaar beoordelen

Frits Feldbrugge
ICA - Informatica Communicatie Academie
Hogeschool van Arnhem en Nijmegen

Samenvatting
Bij ICA en de Faculteit Techniek van de Hogeschool van Arnhem en Nijmegen is er inmiddels enkele jaren ervaring met groepsgewijze beoordeling van studenten door elkaar (peer assessment). Deze wijze van beoordelen wordt mede mogelijk gemaakt en ondersteund door het programma IRIS, dat bij ICA is ontwikkeld. Dit programma, dat ondersteuning biedt bij het invullen en de verwerking van de assessment-gegevens, is reeds gedemonstreerd tijdens NIOC 1999 in Enschede. Inmiddels hebben we veel ervaring opgedaan met peer assessment en het gebruik van IRIS. En het belang ervan neemt alleen maar toe nu het onderwijs steeds meer competentiegericht wordt en de vraag naar methoden en middelen voor assessment toeneemt. Dit artikel beschrijft onze ervaringen tot nu toe.
Keywords: Peer Assessment, assessment, beoordeling, projectonderwijs, probleemgestuurd onderwijs, competentiegericht onderwijs, IRIS

Wat is Peer Assessment?

Peer Assessment (PA) zou je kunnen vertalen als "collegiale beoordeling". Het betekent dat een groep mensen elkaar op bepaalde punten beoordeelt. Mits goed toegepast kan dit een veel beter beeld opleveren dan wanneer een externe observator (docent, chef) een oordeel geeft.

Waar kun je PA toepassen?

PA kun je overal inzetten waar sprake is van groepen mensen. In het onderwijs wordt veel in groepen gewerkt, onder meer bij onderwijsvormen als projectonderwijs en probleemgestuurd onderwijs. In de meeste bedrijven en organisaties werken de medewerkers ook in groepsverband: in projectgroepen, afdelingen of units.

Je past PA toe als je waarde hecht aan het oordeel van de groep waartoe je behoort of waaraan je leiding geeft. Zo kan PA gebruikt worden om groepsleden een spiegel voor te houden wat betreft hun inbreng, hun klantgerichtheid, hun communicatieve eigenschappen, hun vakbekwaamheid en noem maar op. Het kan de groepsleden helpen hun sterke en zwakke plekken te ontdekken (in de ogen van hun collega's) en daar hun voordeel mee te doen. Een leidinggevende kan PA natuurlijk ook toepassen om te bepalen hoe de medewerkers optimaal kunnen worden ingezet of anderszins bestuurlijke beslissingen te ondersteunen.

PA-stappenplan

Als je PA wilt toepassen, moeten achtereenvolgens de volgende stappen worden uitgevoerd:

1. ***Doel vaststellen***
   Wat beoog je met een PA?
2. ***Items vaststellen***
   Wat wil je meten met het PA?
3. ***Beoordelingsschaal vaststellen***
   Hoe groot is de beoordelingsschaal en wat is de betekenis ervan?
4. ***PA uitvoeren***
   Alle betrokkenen voeren het PA uit door voor alle collega's per item een oordeel in te vullen.
5. ***Resultaten produceren***

Per betrokkene wordt een beoordelingsoverzicht geproduceerd op grond van de beoordeling door alle andere collega's.

6. ***Resultaten verwerken***
   De beoordelingsresultaten worden verwerkt overeenkomstig het doel van het PA. Zo wordt er bijvoorbeeld een nabespreking met alle betrokkenen gehouden.

Onze ervaringen met PA

Bij ICA, de Informatica Communicatie Academie van de Hogeschool Arnhem en Nijmegen, wordt PA al ruim vijf jaar toegepast bij projectonderwijs. We zullen per onderdeel van het PA stappenplan aangeven welke keuzes wij gemaakt hebben en welke adviezen wij kunnen geven op grond van onze ervaringen.

1. ***Doel PA***
   Het doel van PA in ons projectonderwijs is tweeërlei:
   a. Het elkaar voorhouden van een spiegel, zodat de student weet hoe hij op de betreffende items overkomt naar anderen toe. Hij kan dit ook vergelijken met zijn zelfbeeld. Dit ondersteunt het proces van zelfreflectie en maakt ook mogelijke verbeterpunten zichtbaar.
   b. Het uitfilteren van meelifters. Dat gebeurt door een speciaal daarop toegesneden categorie van items.

   Advies:
   - *Het doel van PA moet goed gecommuniceerd worden naar de betrokkenen.*
     Als het belang van PA niet duidelijk is, is er ook weinig bereidheid om er serieus aan deel te nemen.

2. ***Items***
   Welke vragen moet je stellen? Dat hangt helemaal af van het doel van een PA op welke onderdelen je elkaar beoordeelt.
   Momenteel wordt bij ICA een lijst van ruim 20 items gehanteerd. Per project wordt bepaald, welke items daar een rol spelen. Bij ICA wordt in het eerste project een kortere lijst gehanteerd, bij het tweede project komt daar wat bij en vanaf het derde project wordt de volledige lijst gehanteerd. Dat kan natuurlijk ook anders. Je kunt voor elke onderwijseenheid (project, course, ...) bepalen of je daar PA wilt toepassen of niet en zo ja bepalen welke PA-items je daar wilt hanteren. Dat kunnen bijvoorbeeld de items zijn die betrekking hebben op trainingen, die in het kader van die onderwijseenheid gegeven worden.
   Om "meelifters" uit te filteren uit een projectgroep, is een speciale categorie items, genaamd "projectbijdrage" in het leven geroepen met de volgende items:
   - Is alle projecturen aanwezig.
   - Besteedt projecturen aan het project.
   - Houdt zich aan gemaakte afspraken.

   Als een student aan het eind van een project negatief scoort op een of meer van deze drie items, dan haalt hij het project niet.
   De overige items zijn bedoeld om de student een spiegel voor te houden en bevat momenteel items op het gebied van:
   - Zelfstandigheid
   - Spreekvaardigheid
   - Assertiviteit
   - enz.

   Adviezen:
   - *De lijst moet niet te lang zijn.*
     Te lange lijsten vergroten de weerstand om de lijst in te vullen.

Men maakt zich er dan snel van af, zodat de meting aan waarde inboet. We duiden dit aan met de term "PA-moeheid". Een lijst van ten hoogste 15 items lijkt ons goed werkbaar.
Eigenlijk vinden we de huidige ICA-lijst met zijn ruim 20 items te lang. Dit zal binnenkort nog eens kritisch worden bekeken.

- *Elk item moet helder geformuleerd zijn.*
  Het moet van meet af aan duidelijk zijn wat er wordt bedoeld. Ook de bijbehorende omschrijving kan helpen om de bedoeling nog duidelijker te maken.
- *Elk item moet meetbaar of waarneembaar zijn.*
  De beoordelaar moet zich een duidelijk beeld kunnen vormen, waar de beoordeelde staat met betrekking tot het item. Liefst moet het oordeel kunnen worden onderbouwd met concrete voorbeelden.
- *Zorg ervoor dat alle items positief zijn geformuleerd, d.w.z. dat de hoogste score de meest wenselijke is.*
  Voorbeelden:
  goed: Zelfstandigheid
  fout: Slordigheid
  Als alles positief is geformuleerd, zie je in één oogopslag welke items mogelijke verbeterpunten zijn.

3. Beoordelingsschaal

Momenteel wordt bij ICA een schaal van -- tot ++ gehanteerd, die overeenkomst vertoont met de symbolen zoals we die kennen van de Consumentenbond. Daarbij hoort de volgende betekenis:

++ zeer goed, echt een heel sterk punt
\+ duidelijk beter dan gemiddeld
0 gewoon, gemiddeld, voldoende
\- duidelijk minder dan gemiddeld
-- heel slecht, duidelijk een heel zwak punt

Advies:

- *De betekenis van de schaal moet duidelijk gecommuniceerd zijn naar alle betrokkenen.*
  Als de betekenis niet duidelijk is, zijn de beoordelingen niet vergelijkbaar. Let er bijvoorbeeld op dat "0" een gemiddeld niveau aangeeft en dus niet noodzakelijkerwijs een verbeterpunt aangeeft.

4. PA uitvoeren

Bij de uitvoering van het PA wordt gebruik gemaakt van het programma IRIS, dat bij ICA is ontwikkeld en momenteel op diverse plaatsen binnen de hogeschool in gebruik is. De tutor zet de PA-gegevens klaar voor de studentengroep. Daarna vullen de studenten hun PA-beoordelingen in.

Adviezen:

- *Overtuig de studenten ervan dat ze hun groepsgenoten eerlijk moeten beoordelen.*
  Ze moeten niet bang zijn een "matennaaier" te zijn. Wat betreft meelifters heeft niemand er baat bij als studenten hun project halen, terwijl anderen de kastanjes uit het vuur hebben gehaald. Wat betreft het "spiegel voorhouden" onthoud je iemand feedback die kan leiden tot verbeterpunten, als je hem te positief beoordeelt.
  Onze ervaring is dat propedeuse-studenten nog wel eens moeite hebben om hun collega's negatief te beoordelen. Maar hogerejaars studenten doen het steeds beter en eerlijker.

- *Let op de betekenis van de scoreschaal.*
  Een "0" betekent "gemiddeld". Geef niet te gemakkelijk plusjes. Alleen als iemand bovengemiddeld is. En minnetjes als iemand ondergemiddeld is.
  Onze ervaring is dat veel propedeuse-studenten elkaar te positief beoordelen. Als je hen daarop wijst, wordt het oordeel realistischer.
- *Let op dat de beoordelingen los van elkaar staan.*
  Het komt nog wel eens voor dat een student een ander zo negatief vindt dat hij hem op alle items met minnetjes "beloont", ook op punten waar misschien zelfs een plusje zou moeten staan. M.a.w. studenten moeten leren zo objectief mogelijk te kijken naar hun groepsgenoten in relatie tot het te beoordelen item.
- *Geen oordeel? Niet invullen!*
  Niet alle items kun je beoordelen. Want je kent niet van iedere student alle facetten. Zeker als je nog niet zo lang met elkaar optrekt. Als een student geen goed beeld heeft van een ander, vult hij soms een "0" in. Dat is onterecht, want een "0" is wel degelijk een oordeel. Benadruk dat de student in dat geval "geen oordeel" invult, anders verstoort hij onbedoeld de meting.
- *Zorg ervoor dat studenten hun beoordeling kunnen onderbouwen.*
  De student moet kunnen motiveren waarom de betreffende beoordeling gegeven is. Een vaag gevoel is niet voldoende.
- *Pas op voor beoordeling van incidenten.*
  De beoordeling dient gebaseerd te zijn op een stabiel beeld. Dat een student zich laatst een keer verslapen heeft wil niet zeggen dat hij altijd te laat komt.
- *Check of student zich aan de meting onttrekt of zich ervan af maakt.*
  Het kan zijn dat een student niets heeft ingevuld. Als hij bij alle collega's alle items als "gemiddeld" heeft ingevuld is ook de kans groot dat hij zich ervan af heeft gemaakt. In het laatste geval kun je de student beter niet mee laten wegen in de berekening van het eindresultaat. In beide gevallen is het raadzaam de student ter verantwoording te roepen en het belang van de PA-meting nog eens duidelijk te maken.

5. Resultaten produceren

Hier bewijst het programma IRIS vooral zijn nut. Als je een PA op papier zou uitvoeren, zou je al gauw merken dat de verwerking erg arbeidsintensief is, zeg maar gerust: monnikenwerk. Immers, elk groepslid beoordeelt elk ander groepslid op een aantal onderdelen. Als je een groep van 10 personen hebt en iedereen gebruikt één formulier voor de beoordeling van elke collega, dan heb je in totaal 90 formulieren te verwerken. Maar zelfs bij kleinere groepen van 5 personen gaat het verwerken van 20 formulieren op den duur tegenstaan.
IRIS is bedoeld om dit monnikenwerk te verlichten.
Wij houden per project (duur: een semester) vier PA-rondes; daarvan is de eerste en derde ronde een "korte", waarbij alleen de drie items betreffende de "projectbijdrage" worden beoordeeld. Op deze wijze kan een student tijdig gewaarschuwd worden; immers als hij op een van deze items aan het eind negatief scoort, haalt hij het project niet.

De tweede en vierde ronde zijn complete rondes, waarbij alle items worden ingevuld en achteraf in de groep worden besproken.
Binnen ICA hebben wij inmiddels meer dan vijf jaar ervaring met het gebruik van IRIS. Het is een stabiel product dat zich bewezen heeft. Het programma is tegenwoordig ook voor andere onderwijsinstellingen tegen een redelijke vergoeding beschikbaar.

Adviezen:

- *Houdt niet teveel PA-rondes.*
  Net als te lange lijsten vergroten teveel PA-rondes de kans op "PA-moeheid".
- *Het invoeren van PA in de onderwijs-organisatie behelst meer dan alleen het aanschaffen van het IRIS-programma.*
  Met IRIS heb je nog geen Peer Assessment in huis. Het goed toepassen van PA vereist, dat de organisatie daarop is toegesneden. Het moet bekend zijn, waarom PA wordt toegepast en hoe. Wie daarbij welke rollen speelt. IRIS moet slechts gezien worden als een ondersteunend gereedschap, dat PA uitvoerbaar maakt.

6. Resultaten verwerken

Aan het eind van iedere PA-ronde wordt er een nabespreking met de projectgroep gehouden. De studenten gebruiken de resultaten onder meer in hun zelfreflectieverslagen. Daarnaast blijkt dat tijdens de nabespreking ook vaak groepsprocessen aan de orde komen.

Advies:

- *Besteed voldoende aandacht aan de nabespreking.*
  Als de verwerking van de resultaten niet goed is, wordt het nut van PA teniet gedaan.
- *Besteed niet alleen aandacht aan de negatieve scores.*
  Als docenten zijn we vaak geneigd alleen de nadruk te leggen op negatieve scores. Studenten vinden het echter ook fijn om te horen waar ze bovengemiddeld scoren en waarom. Zo leren ze ook hun sterke punten kennen.
- *Beschouw negatieve scores niet alleen als negatief.*
  Negatieve scores vormen vaak indicaties van mogelijke verbeterpunten. Het positieve nieuws is dat studenten hierdoor inzicht krijgen in mogelijke leerdoelen voor het verdere leertraject.

PA in competentiegericht onderwijs

Inmiddels wordt op onze hogeschool competentiegericht onderwijs in de breedte ingevoerd, dus ook bij ICA. Een van de grote vraagstukken daarbij is, hoe de competenties beoordeeld moeten worden. Zeker nu het onderwijs steeds extensiever wordt en veel nadruk ligt op het "leren leren", is het aantal contactmomenten tussen docent en student aan het afnemen.
PA kan in het assessmenttraject een belangrijke rol gaan spelen. Er zijn veel competenties, waarbij studenten onderling van elkaar een veel beter en betrouwbaarder beeld hebben dan de docenten.
Natuurlijk zijn er beren op de weg. Je wilt er zeker van zijn dat studenten elkaar niet "matsen". Maar wij zijn ervan overtuigd dat PA een belangrijke rol kan spelen in het vullen van portfolio's van studenten, waarmee zij bewijslast kunnen aandragen inzake het beheersen van bepaalde competenties.

Conclusies

Onze ervaringen met Peer Assessment zijn zonder meer positief en het is ook niet meer uit ons onderwijs weg te denken. In de loop der jaren hebben we de valkuilen leren kennen en weten daar nu goed mee om te gaan.

Nu wij ons onderwijs meer competentiegericht maken verwachten wij PA ook goed te kunnen inzetten bij de competentie-assessments, waarbij studenten PA-resultaten in hun portfolio kunnen opnemen.

ICT-onderwijs in een internationale omgeving

Ludo Kockelkorn, Hogeschool Zuyd

Globalisering is een van de trends die van grote invloed zijn op de ICT-sector in de ruime zin van het woord. Maar is weinig bekend welke effecten dit zal hebben op het hoger ICT-onderwijs en hoe docenten hierop kunnen inspelen. Ludo Kockelkorn probeert een denkrichting te formuleren en houdt een pleidooi voor meer aandacht voor internationale aspecten binnen het ICT-onderwijs. Tijdens het NIOC te Groningen hoopt hij een samen met andere congresgangers de gedachten aan te scherpen.

Context

In India traint men medewerkers van call centra op de diverse Anglosaksische accenten waardoor de klant geen idee heeft, dat hij door iemand aan de andere kant van de aardbol wordt geholpen. De tijd van Jules Verne waarin een reis rond de wereld 80 dagen kostte is allengs voorbij. Bij fysieke goederen is transport steeds een aanzienlijke tijd- en kostenfactor, maar bij digitale producten en diensten naderen deze het nulpunt. De ICT-omgeving verandert in een hoog tempo door offshoring: het tegen lagere kosten onderbrengen van het ontwerp, de realisatie en het beheer van (informatie)systemen in landen als India, China, Rusland en andere landen. Call centra en andere service diensten worden in drie tijdzones van (3x8 uur) ingericht en met name de Engelstalige organisaties profiteren daarvan.

Het Nederlands hoger onderwijs houdt deze globale ontwikkelingen slecht bij en er wordt slechts een klein deel hiervan in de Engelse taal verzorgd. Het Nederlandse ICT-onderwijs is helemaal niet ingericht op internationale omgevingen en zeker niet op het outsourcen van ICT naar lagelonen landen en het werken in drie tijdzones. Hogescholen hebben mijn inziens hun internationaliseringsbeleid maar zeer ten dele geïmplementeerd en op faculteits- of opleidingsniveau is het nog slechter gesteld.

Oogmerk

Mijn intentie is om op een persoonlijke manier aandacht te vragen van docenten en beleidsmakers in het hoger onderwijs voor de effecten van globalisering op het ICT-onderwijs. Bovendien tracht ik impliciet en expliciet een aantal mogelijke toekomst-scenario's te schetsen. Tot slot doe ik een oproep om allianties te vormen om daarmee een vertaalslag te maken van algemeen internationaliseringsbeleid van hoger onderwijs naar concrete verbeteringen. Laat ik beginnen met twee statements die je als onderwijsmanager of docent in de media kunt lezen.

Prikkelende uitspraken

Carla Kiburg (Bestuurder FNV Bondgenoten, ICT-dienstensector)[i] noemt op een FNV congres: *"Offshoring van ICT-werk neemt een grote vlucht. Nu al laten 200 bedrijven automatiseringstaken over aan goedkoop personeel in Azië. Europa loopt 2-3 jaar achter de VS aan, de vraag is of hier niet hetzelfde wordt*

gereageerd. In Hongarije vindt nu al jobhopping plaats: het nieuwe werk verdwijnt zodra het elders weer goedkoper kan.
FNV vreest dat Nederland deze ontwikkeling binnen 5 jaar vijftigduizend banen kost." Uit haar uitspraak blijkt een bezorgdheid om werknemers, economie en het welbevinden van een flink deel van de Nederlandse bevolking. Ook als de inschatting van mevrouw Kiburg maar ten dele uitkomt is het een belangrijk bedreigend signaal naar opleidingen in Nederland.

Wilbert Kieboom (CEO Benelux & Scandinavia Atos Origin) stelt in Management Scope[ii]:
"Daar waar detachering afneemt, neemt outsourcing in versneld tempo toe. Ik heb het liever over global sourcing. Ik vind het spannender om mijn klanten uit te leggen dat ik wereldwijd vijftigduizend resources heb, dan dat ik vertel over een paar duizend mensen in India. De lonen in India stijgen snel. Polen en China zijn ook in de mode. India haalt mensen uit Europa voor kennis en kwaliteit. Het maakt de klant niet uit of er in Portugal of India wordt geproduceerd. Het is belangrijk dat je flexibel bent, talenkennis hebt en snel kunt implementeren." Hieruit blijkt mijn inziens hoe binnen menige grote onderneming gedacht wordt, maar ook gedacht moet worden. Maar de goede verstaander begrijpt dat de heer Kieboom hier ook kansen prijsgeeft voor het ICT-onderwijs in Nederland. De kunst is echter om kansen te benutten zodra ze voorbij komen.

Persoonlijke ervaringen
Om nu een koers vast te stellen is het noodzakelijk om te weten waar je vandaan komt. Als Limburgse teenager heb ik de sluiting van de mijnen meegemaakt; wat door de toenmalige bevolking als een verschrikkelijk ingrijpend sociaal-economisch proces is ervaren. In den beginne hield men het verdwijnen van een complete industrietak zelfs voor onmogelijk. Na mijn afstuderen in de jaren '80 hebben de huidige 40+-ers een fase doorgemaakt met 800.000 werklozen in Nederland. Onze huidige studenten kennen dit soort waarnemingen alleen uit de geschiedenisboeken.

Als docent en als opleidingsmanager heb ik eind jaren '90 volstrekt andere verschijnselen gezien. Door bedrijven, ministeries, in de media en op voorlichtingsmateriaal van scholen werd melding gemaakt van schrikbarende tekorten aan deskundige ICT-medewerkers. De bijna-afgestudeerden werden met premies naar promotie-bijeenkomsten gelokt van ICT-bedrijven om vervolgens met auto, notebook en een arbeidsovereenkomst de zaal te verlaten. De Ministeries van EZ en OCW stimuleerden financieel en met regelgeving om duale leerroutes in te richten. Nu zijn we in sneltrein vaart over de gouden ICT-bergen heen.

De eigen faculteit werkt -zoals in mijn situatie- reeds samen met een Belgische en een Duitse opleiding in het kader van de opleiding Communication and Multi Media Design. Daarin worden binnen een fysieke straal van 50 kilometer de culturele, juridische, didactische, sociale en economische verschillen heel duidelijk. Is die cirkel achteraf niet veel te klein getrokken?

Flexibiliteit als stabiele factor
Global sourcing is voor IBM, Microsoft en Oracle de standaard geworden.

Nederlandse bedrijven moeten ook die kansen benutten en kennisinstellingen dienen een basis te vormen om dat te kunnen realiseren. Een voorbeeld van dergelijk bedrijf is Lizatec, een Nederlands bedrijf met een offshore softwarefabriek in St. Petersburg. Projecten worden uitgevoerd onder Nederlandse projectleiding, de feitelijke bouw en technisch beheer vindt plaats in St. Petersburg. Lizatec [iii] noemt op haar site tien misverstanden over offshore automatisering; drie daarvan zijn:

1. Ze lopen technisch achter;
2. Er hoeft niets te veranderen in mijn organisatie;
3. Het zal zo'n vaart niet lopen.

Deze stellingen stemmen mijn inziens ook tot nadenken voor het Nederlands hoger onderwijs.

Na dualisering moeten we ons nu ondernemend instellen op globalisering, om daarmee te voldoen aan de vragen vanuit het bedrijfsleven en daarvan afgeleid van studenten. Het is essentieel voor studenten om all-round opgeleid te worden. Als ze alleen verstand van bedrijfsprocessen hebben kunnen ze niet communiceren met de technici die offshore aanwezig zijn. En als ze alleen verstand van techniek hebben, dan zitten ze in zwaar weer. In plaats van tekorten aan ICT-ers bestaat het risico van overschotten aan te laag of verkeerd geschoolde ICT-ers in de westerse wereld. Daarop moeten we anticiperen.

Leren in een internationale netwerkomgeving

Maar wat moet je nu doen als onderwijsmanager van een ICT-opleiding in Nederland opererend in een internationale omgeving?

Ten eerste biedt de inrichting van de BaMa-structuur ruimte om een internationaliseringsslag te maken. Binnen de Bacheloropleiding worden naast de Major bovendien Minor programma's ontwikkeld die kansen bieden om zo iets als *'offshore automation'* te implementeren. Dit kan relatief snel en betaalbaar een vliegwielfunctie vervullen. Maar veel opleidingen zijn niet alleen te klein om zelfstandig volgende stappen te zetten, maar het docententeam en de studentenpopulatie bestaat uit overwegend Nederlanders. Buitenlandse partnerinstellingen zijn daarom een must bij de volgende operatie. Fysieke uitwisseling van studenten, docenten en studiemateriaal zijn eerste stappen, om vervolgens elektronische leeromgevingen aan elkaar te koppelen en virtuele internationale studententeams in te richten.

De onderwijsmanagers zullen hun aandacht van Den Haag naar Brussel moeten verleggen en zich dienen te verdiepen in EU-subsidies. Die ondersteunen vernieuwingen van elektronische leeromgevingen van instellingen met internationale partners. Op de Online Educa Berlin 2004[iv] wordt aandacht besteed aan deze nieuwe leeromgevingen. Hiermee wordt weer eens aangetoond dat het gebruik van de Engelse taal in het onderwijs essentieel is om kennis met elkaar, in dit geval elektronisch, uit te wisselen. Alleen door deze wegen te volgen kan onze Europese kennismaatschappij in de pas blijven met het tempo in de VS en Japan en de veranderingen in landen als China, India en Rusland.

Het lijkt mij verstandig om train de trainerprogramma's inzake globalisering, offshoring, Engelse taal en inzicht in internationaal ICT-onderwijs in te richten. Uiteindelijk leidt dit tot Engelstalige

onderwijsprogramma's, joint degrees, Engelstalig lesmateriaal en literatuur, cases waarin offshoring aan bod komt, stage- en afstudeeropdrachten in een internationale omgeving en samenwerkingsprojecten met groepen studenten en docenten uit drie tijdzones. Het is een opsomming van suggesties die de door beleidsmakers vertaald moeten worden in concrete uitvoerbare plannen.

Als de gedachten de vrije loop krijgen, zou men zelfs kunnen denken aan het outsourcen van het ontwikkelen van onderwijsmateriaal. Want lopen de internationale onderwijsinstellingen niet enorm achter op het internationale bedrijfsleven, waarbij dit voor een deel gemeengoed is? Want zeg nou eerlijk, wie heeft in Nederland de syllabi van ICT-modules van de *University of Hyderabad*, India, bestudeert? Leren in een internationale netwerkomgeving als competentie verdient mijn inziens van management, docenten en studenten veel meer aandacht te krijgen dan totnogtoe het geval is. Uiteindelijk zal de eigen opleiding en instelling moeten opgaan in een internationaal netwerk, het geen gepaard zal gaan met veel veranderingen en risico's, maar ook boeiende vak-inhoudelijke en loopbaanperspectieven leveren.

Where are we going to?

Dit brengt mij uiteindelijk tot het formuleren van een aantal mogelijke toekomstscenario's voor ICT-opleidingen waarbij de lezer zelf voor- en nadelen mag bedenken:

1. *niets doen* en afwachten; dit is penny wise en pound foolish; maar als de korte termijn gebruikt wordt voor goede plan- en netwerkvorming niet verkeerd.
2. *paniekerig reageren* en zelfstandig een internationaliseringsslag maken; hierbij lopen opleidingen het risico te investeren in dure trips van managers en beleidsmakers en de uitwisseling van enkele studenten, maar blijkt achteraf een richting die niet duurzaam en voldoende grootschalig blijkt te zijn.
3. *nationaal samenwerken* zoeken; via samenwerkingsverbanden met ander hogescholen en/of universiteiten is de internationale slagkracht veel groter; investeringen kunnen over meer m.n. studenten worden uitgesmeerd.
4. *Noord-Atlantisch coöpereren* met Engelstalige partners in Engeland of de VS; dit is een traject dat door veel opleidingen reeds succesvol wordt bewandeld, maar waar partners uit andere werelddelen tot nog toe veelal ontbreken.
5. *globaal netwerken* met externe onderwijs- en bedrijfspartners in drie tijdzones; door scenario 4 uit te breiden met partners (ook bedrijven) in Europa (met name ook de nieuwe lidstaten), Amerika en Azië is het mogelijk om letterlijk 24 uur aan kennisontwikkeling, -deling en –circulatie te doen; het onderwijs volgt daarin multinationale organisaties.

Met dit artikel hoop ik een bijdrage te leveren aan het verder globaliseren van het hoger ICT-onderwijs. Technisch kunnen we binnen enkele seconden internationale contacten leggen, maar nu moeten onder andere ICT-docenten hun eigen techniek ook toepassen en kennis op wereldniveau uitwisselen. Mijn indruk is dat alleen de laatste optie 5 getuigt van een visie op globale ontwikkelingen en gericht is op een duurzame aanpak. Uiteraard zou je daar eisen aan moeten stellen zoals deelname van Aziatische onderwijs-

partners, multinationale organisaties, een Engelstalige elektronische leeromgeving en leeropdrachten die in drietijdzones worden uitgevoerd. In de komende 1 à 2 jaar zullen hogescholen posities (her)overwegen en gewild of ongewild afstevenen op een van de genoemde scenario's. De toekomst zal het uitmaken wie de wijsheid in pacht heeft en de juiste optie heeft gekozen.

Literatuur

Lansbergen, J. (2004), 'Hypes, hopes en hobbels: hoe vergaat het de Engelstalige opleidingen Informatica?', TINFON, 13e jaargang 2004, nr. 2, blz. 53-55
Poort, J. en Zijderveld, C. en Brouwer, N. (2204), Verplaatsing industrie: hoe erg is het?, Stichting voor Economisch Onderzoek der Universiteit van Amsterdam

URL's	Toelichting
www.ictforyourbusiness.nl	Site met de vraagstelling 'maakt outsourcen opleiden in ICT niet overbodig?'
www.lizatec.com	Lizatec is een Nederlands bedrijf met een ontwikkelafdeling in Rusland.
www.nuffic.nl	Netherlands Organization for International Cooperation in Higher Education
www.offshorecm.com	Offshorecm is een website met informatie over offshore automatisering en vele links.

Noten

[i] FNV congres, juni 2004
[ii] Management Scope, juni 2004, p. 40
[iii] www.lizatec.com
[iv] www.surf.nl

Ervaringen met een minor eBusiness in het HBO

Henk Plessius, Hogeschool van Utrecht
Pascal Ravesteyn, Hogeschool van Utrecht

Met de komst van de Bachelor/Master structuur in het hoger onderwijs worden veel curricula herzien. Een nieuw fenomeen daarbij is de invoering van minor-programma´s in het HBO (het WO kende deze structuur al langer). Aan de Hogeschool van Utrecht heeft in het najaar van 2003 een pilot plaatsgevonden van zo´n minor met als thema eBusiness. Deze minor was opgebouwd uit verplichte modulen, keuzemodulen en een (forse) praktijkopdracht voor een bedrijf.
Op basis van de pilot is een gegeneraliseerd raamwerk ontwikkeld voor een (in de tijd) beperkt onderwijsprogramma (typische omvang een half jaar) in het hoger onderwijs.

Keywords: hoger onderwijs, minor, eBusiness, minor-raamwerk

Inleiding

Sinds september 2002 kent Nederland een nieuwe structuur voor het hoger onderwijs: de Bachelor/Master structuur. Hiermee conformeert Nederland zich aan de binnen de EU opgestelde intentieverklaring van Bologna (1999) om ´an open and transparent European Higher Education Area´ te creëren.
De Hogeschool van Utrecht (www.hvu.nl) heeft deze gelegenheid aangegrepen voor curriculum vernieuwing. Een van de grote veranderingen is de invoering van een minor in alle Bachelor-programma´s. Onder een minor wordt daarbij verstaan een samenhangend studieprogramma van 30 ECTS (een half studiejaar). Een minor kan een verdieping zijn van de gekozen opleiding (die in dit verband ook wel major genoemd wordt en een omvang van 3,5 jaar heeft), maar kan ook verbredend zijn en kennis uit een heel ander domein aan de orde stellen. Studenten Informatica bijvoorbeeld kunnen hun profiel verdiepen met een minor Human Computer Interaction of Game Design, maar ook verbreden met een minor Ondernemen of Spaans.
Met de introductie van minor-programma´s beschikt de Hogeschool van Utrecht over een flexibel onderwijssysteem dat studenten de mogelijkheid biedt in behoorlijke mate invloed uit te oefenen op de competenties die ze tijdens hun studie ontwikkelen; in veel opleidingen kan deze flexibiliteit nog verder oplopen (naast de minor zijn bijvoorbeeld ook de stage en het afstuderen redelijk vrije onderdelen, al geldt voor deze laatsten wel dat ze moeten aansluiten bij de major).

In september van dit studiejaar (2004-2005) is de Hogeschool officieel begonnen met de uitrol van minor-programma´s. Een 20-tal minors is dit studiejaar ook daadwerkelijk gestart. Om alvast ervaringen op te doen met het fenomeen minor, heeft vorig studiejaar (2003-2004) een pilot met een minor eBusiness plaatsgevonden. Als doelgroep is gekozen voor studenten van één van de zeven ICT-opleidingen van de Hogeschool (zie textbox).

De Hogeschool van Utrecht biedt via drie verschillende faculteiten zeven verschillende Bachelor-opleidingen aan in het domein van de Informatie- en Communicatietechnologie:
- *Faculteit Natuur en Techniek*: Informatica, Technische Informatica, Mediatechnologie en Information Engineering
- *Faculteit Communicatie en Journalistiek*: Communicatiesystemen en Communication & Multimedia Design
- Faculteit Economie en Management: Bedrijfskundige Informatica

In 2003 is een alignment-programma opgezet om te komen tot meer samenhang tussen deze opleidingen: het Picture-programma (zie www.picture.hvu.nl). De pilot-minor eBusiness is een van de eerste tastbare resultaten van dit programma.

Textbox: de ICT-opleidingen van de Hogeschool van Utrecht

De minor eBusiness

Vanaf het begin heeft vooropgestaan dat het curriculum het terrein van de eBusiness in de breedte moet dekken. Dat betekent aandacht voor zowel bedrijfskundige aspecten (wat is de impact op de organisatie), communicatieve en marketingaspecten als de realisatie van een applicatie. Al deze deskundigheden zijn bij de ICT-opleidingen van de HvU aanwezig en moeten in de context van eBusiness hun plaats krijgen. Daarnaast zijn, om recht te doen aan de verschillen in voorkennis en cultuur, voor de minor eBusiness de volgende uitgangspunten opgesteld:
- er moet in de minor een eBusiness applicatie voor een (externe) organisatie gerealiseerd worden
- om een gemeenschappelijke achtergrond te bereiken, worden enkele verplichte modules opgenomen; om eigen interesses in de eBusiness uit te kunnen diepen, kunnen studenten aanvullend zelf modulen kiezen uit een aanbod
- er moet een professionele werkomgeving beschikbaar zijn (zowel fysiek als virtueel), die het samen-werken stimuleert.

Op basis van deze uitgangspunten is een programma samengesteld. Voor het huidige studiejaar ziet dat er als volgt uit:

Verplichte modulen (12 ECTS):
- *Strategy, change & vision* (3 ECTS): Geeft een overall beeld van de mogelijkheden van E-business, nu en in de toekomst. Wordt volledig verzorgd door externe docenten uit het bedrijfsleven
- *E-commerce* (3 ECTS): Gaat in op de interactie tussen het bedrijf en de klanten: hoe is optimaal te profiteren van Internet-technologie?
- E-procurement en supply chain management (3 ECTS): Onderwerp is de logistiek en de inkoop van bedrijven: hoe doen bedrijven onderling zaken?
- *Organisation & business processes* (3 ECTS): Deze module gaat over de interne organisatie en, in samenhang daarmee, de inrichting van de achterliggende bedrijfs processen.

Keuzemodule (6 ECTS):
Studenten kunnen hun eigen interesses uitdiepen in de keuzemodule. Dit studiejaar worden de volgende onderwerpen expliciet aangeboden: CRM, XML en Project management. Daarnaast kunnen studenten hier ook andere onderdelen uit het aanbod van de HvU opnemen, mits dat in het thema van de eBusiness past (aanschuifonderwijs).

Opdrachten (12 ECTS):
- *E-opdracht* (9 ECTS): deze bestaat uit 2 delen (het ontwerp en de realisatie van een E-business opdracht en de analyse van de verandering door invoering van de applicatie). Deze opdracht wordt door een team van studenten uitgevoerd
- *eScriptie* (3 ECTS): Een (individueel) product over een in overleg met de begeleider vast te stellen deelonderwerp (laat zich goed combineren met de keuzemodule).

Verder beschikken we voor de minor over een royale, goed-geoutileerde, werkruimte waarin studenten in groepsverband aan het werk kunnen zijn. Als virtuele werkruimte wordt gebruik gemaakt van Teletop (www.teletop.nl) dat als eLearning omgeving ook faciliteiten biedt om samen aan produkten te werken.

In de minor zijn studenten overwegend zelfstandig (in groepsverband) aan het werk, waarbij deskundige begeleiders beschikbaar zijn. Het aantal colleges wordt minimaal gehouden (gemiddeld twee dagdelen per week, hoewel sommige keuzemodulen een hogere collegebelasting hebben).

Ervaringen met de minor

De pilotminor die in het najaar 2003 heeft plaatsgevonden, is gevolgd door 38 derdejaarsstudenten van verschillende opleidingen. De pilot is uitgebreid geëvalueerd; de belangrijkste uitkomsten en de wijzigingen die we op grond daarvan hebben doorgevoerd, geven we hier beknopt weer.

Studenten:
De studenten zijn gemiddeld genomen tevreden geweest over de minor, waarbij de studenten Bedrijfskundige Informatica lager dan gemiddeld scoorden – het bedrijfskundige aspect, met name de impact die eBusiness heeft in organisaties, is in de pilot onvoldoende uit de verf gekomen (in het programma van 2004 is daar uiteraard rekening mee gehouden). De minor bleek voor alle studenten goed aan te sluiten op hun vooropleiding en had voldoende theoretische diepgang. De elektronische leeromgeving (Teletop) is buitengewoon goed beoordeeld en heeft waarschijnlijk veel bijgedragen aan het succes van de minor.
Studenten zijn in multi-disciplinaire teams ingedeeld. De projectgoepen hebben over het geheel genomen goed gefunctioneerd, waarbij taakverdeling vooral op basis van deskundigheid heeft plaatsgevonden.

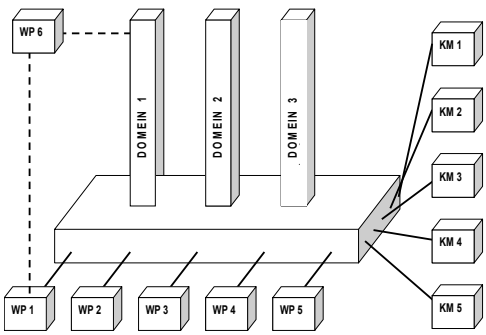
Docenten:
Niet alleen de studenten hebben verschillende achtergrond, ook de docenten komen uit andere faculteiten en daarmee andere culturen. Over de minor in het studiejaar 2003 zijn door de docenten– naast commentaar op de inhoud – vooral opmerkingen gemaakt over de samenhang van het programma; door de korte voorbereidingstijd schortte het daar wel eens aan. Dit studiejaar is de voorbereiding grondiger geweest, maar waarschijnlijk nog belangrijker is het feit dat de docenten in de week voor de start twee dagdelen bij elkaar hebben gezeten en elkaar duidelijk hebben gemaakt welke doelen ze wilden bereiken. Ook is veel aandacht besteed aan een gemeenschappelijk gedragen didactische aanpak. Hierbij is uiteindelijk gekozen voor de Silkstone methode (van Elsen, 2004) gebaseerd op de visie dat studenten leren ´door te werken, niet om te werken´.

Opdrachtgevers:
Het werken met externe opdrachtgevers is altijd een risico: de verwachtingen van opdrachtgever, studentengroep en opleiding

kunnen uiteen lopen. Ook kunnen er inhoudelijk grote verschillen ontstaan tussen de opdrachten qua niveau en omvang. Hoewel de opdrachtgevers unaniem heel tevreden waren met de resultaten van de lichting 2003, is het voortraject dit jaar gewijzigd: studenten moeten expliciet een ´offerte´ uitbrengen, waarin heel nauwkeurig beschreven staat wat de opdrachtgever aan het eind mag verwachten (en omgekeerd, wat de studenten van de opdrachtgever verwachten). In dit offertetraject kunnen de eisen van de opleiding meegenomen worden. Bovendien wordt er zo een resultaatverplichting gecreëerd waarvan we verwachten dat deze motiverend kan werken.

Algemeen minor raamwerk

Op basis van de ervaringen welke zijn opgedaan tijdens de ontwikkeling en de uitvoering van de minor eBusiness is een algemeen model opgesteld voor minors. Dit model (de Bruin et al, 2004) kan gebruikt worden onafhankelijk van de inhoud van een minor.



Het model (zie figuur) gaat uit van één centrale module waar alle competenties die de student binnen de minor dient te verwerven, bij elkaar komen in de vorm van een project. Idealiter zal het centrale project niet bestaan uit een standaard case omschrijving maar zijn dit opdrachten welke voor echte opdrachtgevers worden uitgevoerd. De opdrachtgevers zijn niet alleen bedrijven maar kunnen ook non-profit organisaties zijn of zelfs de eigen onderwijsinstelling; het belangrijkste is dat het op te leveren product / de te leveren dienst meerwaarde voor de opdrachtgever oplevert. Typisch heeft dit project een omvang van rond de 9 ECTS.
Omdat het thema van een minor vaak in verschillende domeinen relevant is, kan er voor gekozen worden om het centrale project binnen een bepaald domein te laten plaatsvinden. Zo kan er bijvoorbeeld voor gekozen worden om alleen projectopdrachten te accepteren binnen de domeinen bouw en gezondheidszorg.

Om gericht te kunnen werken aan deelcompetenties, worden workpackages (WP) aangeboden. Dit kunnen colleges, practica, onderzoeksopdrachten, etc. zijn. In de figuur zijn er vijf getekend (WP1 tot WP5) uitgaande van een gemiddelde omvang van 3 ECTS per WP. Het zesde workpackage (WP6) heeft een bijzondere plaats in het model. In het centrale project wordt namelijk (net zoals in het hedendaagse onderwijs) groepsgewijs aan de opdracht gewerkt. In WP6 wordt de studenten gevraagd hun individuele ontwikkeling aan te tonen, bijvoorbeeld in de vorm van een onderzoek naar de toepassing van een onderwerp in een domein. Zo´n opdracht vormt tevens een voorbereiding voor de laatste fase van de studie: het afstudeeronderzoek.

Als laatste zijn in de figuur een aantal keuzemodulen (KM) opgenomen waarmee de student een eigen profilering kan aanbrengen.

Afsluiting

Naar aanleiding van de ervaringen opgedaan in de pilot van de minor e-business is een algemeen raamwerk voor minors ontwikkeld. Dit raamwerk is hierna gebruikt bij de ontwikkeling van meerdere minors.

Doordat met het raamwerk een minor modulair wordt opgebouwd, is het heel eenvoudig om (onderdelen van) modulen uit te wisselen met andere minors. Zo kan bijvoorbeeld een module die tot het kernprogramma (WP) behoort in de ene minor als keuzemodule (KM) opgenomen worden in een andere minor. Tevens is het model goed toepasbaar om minors te ontwikkelen welke grotendeels via eLearning worden aangeboden. In een dergelijk geval zullen de modules (WP en KM) via een Elektronische Leeromgeving ter beschikking worden gesteld, terwijl het project zorgt voor de nodige samenwerking en contacten tussen studenten en docenten (de Bruin et al, 2004). Het speciale workpackage 6 biedt tenslotte de mogelijkheid om individueel verworven competenties vast te stellen.

Literatuur

G. van Elsen et al. *De Silkstone methode, versie 3.1.* Interne publicatie HvU, 2004

L. de Bruin, H. Plessius, P. Ravesteyn. *eLearning in higher Education: a Casestudy*. Paper presented at eLearning Results 2004, Sestri Levante

De feilbare assessor
Over assessment van competenties

Dr. Wouter Schoonman, Lector Saxion Hogescholen

Samenvatting
Bij assessment van competenties is de assessor de zwakste schakel. In een klein experiment tijdens NIOC 2004 werd dit opnieuw duidelijk. Wat te doen aan de feilbaarheid van de menselijke beoordelaar?

Keywords
Assessment, competenties, assessor, betrouwbaarheid

Copyright
Artikel aangeboden aan: Onderzoek van Onderwijs, januari 2005. Uitgever: Koninklijke Van Gorcum, Postbus 43, 9400AA Assen. www.vangorcum.nl

Inleiding
In het hoger onderwijs en in het bedrijfsleven is assessment een populaire bezigheid. In alle gevallen gaat het om het beoordelen van prestaties van een persoon ten einde een beslissing over die persoon te kunnen nemen. Van die beslissing kan veel afhangen: de student ontvangt studiepunten, de sollicitant krijgt de begeerde baan of de deelnemer ontvangt feedback over verbeterpunten. Het meest kwetsbare onderdeel binnen assessment is de menselijke beoordelaar. De mate waarin hij[i] *een juist oordeel velt, bepaalt de waarde van het assessment. Hoe werkt dat?*

Competenties

Wie assessment zegt, zegt competenties. Hoewel er talloze definities in omloop zijn, is een gangbare omschrijving van een competentie: de mix van kennis, vaardigheden en houding die succesvol beroepsgedrag mogelijk maakt (Schoonman, 2004a). Er zijn honderden competenties geformuleerd, waarbij in het Hoger onderwijs vaak een onderscheid gemaakt wordt tussen generieke en beroepsspecifieke competenties. Een voorbeeld van een beroepsspecifieke competentie is het kunnen *programmeren in een bepaalde computertaal*, een generieke competentie is bijvoorbeeld *het voeren van een klantgesprek*. Voordat een assessment kan plaatsvinden moeten de te meten competenties vastgesteld zijn en moeten gedragsindicatoren bekend zijn: op welk gedrag moet de assessor letten en hoe moet bepaald gedrag gewaardeerd worden. Daarna wordt een levensechte beroepssituatie gecreëerd, waar de kandidaat het bedoelde gedrag kan laten zien.

Klantgesprek

Binnen Saxion worden hulpmiddelen en instrumenten op het gebied van Assessment ontwikkeld. De Kenniskring en het Lectoraat Asssessment richt zich daarbij in eerste instantie op gedragsproeven gericht op generieke hbo-competenties. Een van de hulpmiddelen is de simulatie van een Klantgesprek. Het is een sterk vereen-

voudigde oefening – dus ook bruikbaar bij lagerejaars studenten) waarbij drie competenties worden gemeten: Klantgerichtheid, Gespreksvaardigheden en Commerciële vaardigheden. Deze competenties zijn in dit geval ontleend aan het curriculum van de opleiding Informatica binnen Saxion, maar komen in diverse curricula ook voor. Voor de eenvoud zijn per competentie slechts drie gedragingen beschreven. De deelnemers (assessoren) worden geïnstrueerd om als volgt te werk te gaan.

- Bestudeer het formulier en maak je het bedoelde gedrag eigen
- Kijk naar de filmpjes en noteer wanneer je bedoeld gedrag ziet
- Maak nog geen oordeel over de kwaliteit van het gedrag
- Beoordeel na de oefening het waargenomen gedrag als +, 0 of –
- Tel daarna het totaal op

Het onderscheid tussen Observeren en Evalueren is van groot belang (Altink et al, 2004). Zonder dit onderscheid is feitelijk geen sprake van assessment (in de zin van gedragsproeven). Het ontwikkelde formulier bij deze gedragsproef heeft de volgende vorm:

Indicatoren	Gezien	Kwaliteit
		+ 0 -
Klantgerichtheid		
Verplaatst zich in de positie van de klant		
Staat open voor de problemen van de klant		
Formuleert het probleem in termen van de klant		
Gespreksvaardigheden		
Heeft luisterende houding		
Maakt samenvattingen		
Stelt open vragen		
Commerciële vaardigheden		
Vertaalt probleem naar een oplossing		
Doet voorstellen voor verdere samenwerking		
Herhaalt gemaakte afspraken		
Totaal		

Figuur 1. Observatie- en beoordelingsformulier Klantgesprek

In de twee filmpjes (Schoonman, 2004b) is een ICT-er en een klant te zien. Zij voeren een gesprek over een opgeleverde applicatie door de

Figuur 2. Fragment uit 'Klantgesprek'

organisatie van de ICT-er. De klant heeft een aantal kritiekpunten zoals die in de afgelopen periode boven water zijn gekomen. In het eerste gesprek reageert de ICT-er afwijzend en verdedigend en lijkt niet geïnteresseerd in de problemen van de klant. In het tweede gesprek gaat het een stuk beter en worden constructieve en positieve oplossingen voor (dezelfde) problemen bereikt. De beide filmpjes zijn bedoeld als oefen- en demonstratiemateriaal.

Workshop

Tijdens het tweejaarlijkse NIOC congres van 2004 (een organisatie van informatica docenten in het hoger onderwijs) is een kleine assessment workshop gehouden. De oefening Klantgesprek werd vertoond en aan de deelnemers werd de mogelijkheid geboden het gedrag van de ICT-er via het formulier uit Figuur 1 te beoordelen. Dit na de instructie zoals die bij deze (en elke) assessment oefening hoort. De deelname was anoniem en vrijwillig. Uiteindelijk werden negen volledige beoordelingsformulieren verkregen.

Observaties

Als eerste kijken we hoe de negen beoordelaars (beo1 tot en met beo9) de negen gedragingen (indicatoren) in beide gesprekken al dan niet hebben waargenomen (zonder oordeel). De keuze was wel of niet waargenomen (1 of 0). In onderstaande tabellen staan de resultaten:

Gesprek 1	**beo1**	**beo2**	**beo3**	**beo4**	**beo5**	**beo6**	**beo7**	**beo8**	**beo9**
verpl pos	0	0	0	0	0	0	0	0	0
staat open	1	0	0	0	0	0	1	0	1
form klant	0	0	0	0	0	0	0	0	0
luistert	1	0	1	1	1	0	0	1	0
samenvat	0	0	0	0	0	0	0	0	0
open vr	0	0	0	0	0	0	0	0	0
probl opl	1	1	1	1	1	1	1	1	1
samenw	0	0	0	0	0	0	0	0	0
herh afspr	0	0	0	0	0	0	0	0	0
totaal	3	1	2	2	2	1	2	2	2

Tabel 1. Observaties van negen beoordelaars van negen gedragingen (Gesprek 1)

Gesprek 2	**beo1**	**beo2**	**beo3**	**beo4**	**beo5**	**beo6**	**beo7**	**beo8**	**beo9**
verpl pos	0	1	1	1	1	1	0	0	1
staat open	1	1	1	1	1	1	0	1	1
form klant	1	1	1	1	1	1	1	1	1
luistert	1	1	1	1	1	1	1	1	1
samenvat	0	0	0	0	1	0	0	0	0
open vr	1	1	1	1	1	0	1	1	1
probl opl	1	1	1	1	1	1	1	1	1
samenw	1	1	1	1	1	1	1	1	1
herh afspr	0	0	0	0	1	0	0	0	0
totaal	6	7	7	7	9	6	5	6	7

Tabel 2. Observaties van negen beoordelaars van negen gedragingen (Gesprek 2)

In de rijen staan de negen gedragsindicatoren uit het formulier van Figuur 1. In de kolommen staan de beoordelaars. Een eerste conclusie is dat de negen beoordelaars het *globaal* eens zijn over welke gedragingen te zien waren in gesprek 1 en gesprek 2. In gesprek 1 worden door elke assessor zo'n twee gedragingen geobserveerd, in gesprek 2 zijn dat er ruim zes. In gesprek 1 is de overeenstemming deels artificieel. Het gesprek was zo gemaakt dat het bedoelde gedrag niet of nauwelijks voorkwam. Bij de observaties van gesprek 2 – waar het bedoelde gedrag wel zichtbaar was – bestaan grotere verschillen tussen de beoordelaars. De waargenomen gedragingen lopen uiteen van 5 tot 9 (van de maximaal 9). Het vervelende is dat wanneer een assessor bepaald gedrag niet heeft waargenomen, dat er dan ook geen waardering van dat gedrag kan plaatsvinden. Iets dat er niet is kun je niet waarderen.

Gesprek 1	**beo1**	**beo2**	**beo3**	**beo4**	**beo5**	**beo6**	**beo7**	**beo8**	**beo9**
verpl pos									
staat open	0						-1		-1
form klant									
luistert	1		-1	-1				0	
samenvat									
open vr									
probl opl	0		-1	-1				1	-1
samenw	-1						0		
herh afspr									
totaal	0	0	-2	-2	0	0	-1	1	-2

Tabel 3. Evaluatie van negen beoordelaars van gedrag (Gesprek 1)

Gesprek 2	**beo1**	**beo2**	**beo3**	**beo4**	**beo5**	**beo6**	**beo7**	**beo8**	**beo9**
verpl pos		1	1	1	1	1			1
staat open	1	1	1	1	1	1		1	1
form klant	1	1	1	1	1	1	0	0	1
luistert	1	1	1	1	1	1	1	1	1
samenvat					1				
open vr	1	0	1	1	1		0	1	1
probl opl	1	1	1	1	1	1	1	1	1
samenw	1	0	1		1	1	0	1	1
herh afspr					1				
totaal	6	5	7	6	9	6	2	5	7

Tabel 4. Evaluatie van negen beoordelaars van gedrag (Gesprek 2)

Evaluatie

Conform de instructie is dit assessment proces in twee delen gesplitst: observatie en evaluatie. In het formulier (zie Figuur 1) zijn de rechter kolommen na elkaar en niet gelijktijdig ingevuld. Bij de evaluatie hadden de assessoren de keuze uit +, 0 of -. De betekenis was: positief, neutraal en negatief. Anders gezegd: het zichtbare, geobserveerde gedrag wordt van een waardeoordeel voorzien. In onderstaande tabellen wordt de waardering van de assessoren van *het waargenomen gedrag* weergegeven. Hierbij is een positieve waardering een +1, een neutrale waardering een 0 en een negatieve waardering een -1. In de kolommen staan evaluaties van de negen beoordelaars.

De evaluatie van de assessoren van Gesprek 1 (Tabel 3) is conform verwachting. Er is weinig wenselijk gedrag waargenomen en de beoordeling van wel waargenomen gedrag is overwegend negatief. De gemiddelde evaluatie is -1 wat als 'onvoldoende' vertaald kan worden. Bij de evaluatie van Gesprek 2 wordt veel gedrag positief gewaardeerd, maar de uiteindelijke waardering (laatste rij in Tabel 4) loopt sterk uiteen. Beoordelaar 7 komt tot 2 punten, terwijl beoordelaar 5 het maximum van 9 punten uitdeelt. De beoordelaars zijn het dus eens over zwak gedrag (het bedoelde gedrag wordt niet gezien en voor zover aanwezig negatief beoordeeld), maar verschillen nogal over wat als 'sterk gedrag' moet worden gezien. Voor de kandidaten over wie zo'n oordeel wordt uitgesproken, blijkt het oordeel van een toevallige individuele assessor af te hangen (zie ook Hofstee, 1999). Wanneer assessor 5 een oordeel uitspreekt scoort de kandidaat een '9', wanneer echter assessor 7 dit doet, wordt het een '2'. Hoewel het niet duidelijk is wat een '2' of een '9' betekent, is de discrepantie te hoog. Het mag niet zo zijn dat een oordeel afhankelijk is van een toevallige assessor.

Analyse

De vraag is natuurlijk hoe het zit met de kwaliteit van deze assessment procedure bij de oefening 'Klantgesprek'. Omdat deze oefening is gebaseerd op vastgestelde competenties binnen een bestaande opleiding, lijkt het realiteitsgehalte niet het probleem. De vraag naar het realisme van de onderscheiden competenties en bijbehorende gedragsindicatoren, lijkt niet aan de orde. Wel is het zo dat er in dit experiment forse verschillen blijken op te treden tussen assessoren. De een ziet niet wat de ander ziet. In dit geval is er duidelijk sprake van een fors verschil tussen bijvoorbeeld beoordelaar5 en beoordelaar7. De gemiddelde correlatie tussen de beoordelaars wat betreft waargenomen gedrag is relatief hoog (een correlatie van ongeveer 0.7). De *evaluatie* door de diverse beoordelaars loopt echter sterk uiteen. Het lijkt erop dat zelfs een eenvoudige gedragsproef na instructie van de assessoren niet automatisch leidt tot een uniform oordeel. De individuele assessor lijkt een (te) belangrijke rol te hebben bij de totstandkoming van het uiteindelijke oordeel. Dit lijkt om diverse redenen onwenselijk.

Conclusie

Het is in dit (kleine) experiment opnieuw duidelijk geworden dat de individuele assessor de zwakste schakel is bij assessment. Er bestaan verschillen tussen assessoren, zowel wat betreft het *observeren* van gedrag als het *evalueren* daarvan. Dit leidt in de praktijk tot willekeurige en niet gestandaardiseerde beoordelingen. Het trainen en certificeren van assessoren lijkt daarom de aangewezen weg om tot eerlijke

en professionele beoordeling van beroepsprestaties te komen.

Referenties

Altink, W., Schoonman, W., Seegers, J. (2004). *Menselijk kapitaal. De ontwikkeling van mensen in organisaties*. Assen: Van Gorcum

Hofstee, W.K.B. (1999). *Principes van beoordeling. Methodiek en ethiek van selectie, examinering en evaluatie*. Lisse: Swets & Zeitlinger

Schoonman, W. (2004a). *Assessment voor en door iedereen. Lectorale rede*. Enschede: Saxion Hogescholen

Schoonman, W. (2004b). *Assessment oefening: het ICT klantgesprek*. DVD + formulier. Enschede: Saxion Hogescholen

Dank

Met dank aan Drs. Piet Hendriks – lid van de Kenniskring Assessment van Saxion -voor zijn commentaar op een eerdere versie van dit artikel.

Noten

[1] Leest u ook 'zij' waar 'hij' staat. Over de sexe-verdeling bij assessoren is mij niets bekend.

De Bachelor of ICT, een competentiegerichte profielbeschrijving

René Tönissen, Hogeschool van Amsterdam, instituut voor Informatica

Samenvatting
Het HBO-I heeft in nauwe samenspraak met het bedrijfsleven in de volle breedte van de ICT-branche een nieuw competentiegericht profiel voor de bachelor of ICT samengesteld. Dit document zal, na formele goedkeuring door de HBO-raad, de officiële beschrijving zijn van de domeincompetenties voor de bachelor of ICT. Het is een innovatief document geworden, met een beperkt aantal generieke competenties, die geïllustreerd worden door beschrijvingen van concrete beroepssituaties waarin de nét afgestudeerde bachelor of ICT terecht kan komen.

Keywords
Competentieprofiel, domeincompetenties, bachelor of ICT, bachelor, bama, ICT, HBO, HBO-I

Veranderingen

Sinds de 'Bologna-verklaring' [1] is het landschap van het hoger onderwijs in Nederland ingrijpend veranderd. Vooral de invoering van het bachelor-mastermodel zorgde ervoor dat het veelgebruikte profielenboekje [2] aan vervanging toe was. Met de introductie van vier bachelordomeinen in de sector Techniek/ICT – engineering, applied science, built environment en ICT – ontstond direct de behoefte aan een beschrijving van deze domeinen. Medio 2003 heeft de HBO-raad een proces geïnitieerd met als doel om van beroeps- en opleidingskwalificaties te komen tot raamwerken voor competentieprofielen voor bacheloropleidingen. Het doel van het raamwerk is tweeledig:

- aangeven wat de gemeenschappelijke competenties zijn van studenten die met een bepaalde bachelortitel afstuderen
- een startpunt bieden voor de accreditatie om opleidingen te kunnen beoordelen als ze een bepaalde bachelorgraad afgeven.

De HBO-raad wil al deze competentieprofielen beschikbaar maken op zijn website domeincompetenties (www.hbo-raad.nl). Het HBO-I platform, het overleg van opleidingsmanagers van (bijna) alle hbo-ict-opleidingen in Nederland, heeft als reactie op dat initiatief in september 2003 een werkgroep ingesteld om het competentieprofiel van de bachelor of ICT te beschrijven. Met de oplevering van dit document heeft de bachelor of ICT nu als eerste van de vier genoemde domeinen zo'n beschrijving. Het profieldocument richt zich op een aantal doelgroepen. Ten eerste op het werkveld, dat met dit document een beschrijving krijgt van de net afgestudeerde bachelor of ICT. Ten tweede op de hbo-ict-opleidingen, die dit document kunnen gebruiken bij het ontwerpen van hun leerplannen, maar het ook kunnen inzetten als referentiepunt in het accreditatietraject. Ten derde op hbo-ict-studenten, die het profiel kunnen gebruiken om tijdens de studie richting te geven aan de eigen competentieontwikkeling. En ten vierde

op studieadviseurs, decanen en aspirant studenten, om een idee te krijgen over wat voor kwaliteiten een bachelor of ICT moet beschikken. Daarnaast kan het document wellicht een inspiratiebron voor andere sectoren en/of clusters zijn, in het proces van totstandkoming van competentiebeschrijvingen van andere bachelorprofielen.

Werkwijze van de werkgroep

De werkgroep heeft het 'oude' profielenboekje als uitgangspunt genomen. Van de gedetailleerde opleidingskwalificaties die daarin voor de opleidingen Bedrijfskundige Informatica (BI), (voorheen: Hogere) Informatica (I) en Technische Informatica (TI) waren beschreven is de gemeenschappelijke basis geformuleerd. Dit heeft geleid tot de formulering van tien algemene bouwstenen voor competenties en vijf, voor de ICT specifieke bouwstenen. Voor de algemene bouwstenen heeft de werkgroep gekozen om aan te sluiten bij de tien generieke hbo-kernkwalificaties zoals deze door de commissie Franssen zijn opgesteld, en die in hbo-land inmiddels breed ingang hebben gevonden. De specifieke bouwstenen voor ict-competenties sluiten aan bij de levenscyclus van ict-systemen: analyseren, ontwerpen, realiseren en beheren. Daaraan is de bouwsteen adviseren toegevoegd. De tekstvoorstellen van de werkgroep zijn besproken in een reviewgroep van het HBO-I, en in later stadium besproken en goedgekeurd, door het voltallige HBO-I-platform, het overleg van alle hbo-ict-opleidingsmanagers. De goedgekeurde concepttekst is voorgelegd aan een grote vertegenwoordiging van het werkveld. Meer dan 60 bedrijven, samen representatief voor het volledige spectrum van het ict-domein hebben in een schriftelijke en/of mondelinge reactie de conceptversie becommentarieerd. Dit heeft geleid tot een aantal substantiële wijzigingen in de tekst, en uiteindelijk tot de validering van het document door het werkveld.

Illustraties

Met opzet wordt over bouwstenen voor competenties gesproken. Voor het verwerven van competenties heeft de ict-student een context nodig. Alleen in een professionele omgeving leiden de verschillende bouwstenen tot competent gedrag. De formulering van de (tien plus vijf) bouwstenen voor competenties levert een generieke beschrijving op voor de 'ingrediënten' van de kwaliteiten van elke bachelor of ICT. Het geeft een minimale set van eisen die het werkveld aan de bachelor of ICT mag stellen. Zo'n beschrijving heeft onvermijdelijk een hoog abstractieniveau. Om deze generieke beschrijving kleur te geven en te concretiseren heeft de werkgroep ervoor gekozen om een aantal 'illustraties' toe te voegen van specifieke beroepssituaties waarin een net afgestudeerde bachelor of ICT terecht kan komen. In de illustraties staat telkens één van de specifieke bouwstenen centraal en elke specifieke bouwsteen is in verschillende contexten geplaatst. Zo is er een beschrijving van een beroepssituatie waarin de (junior-) ict'er vooral wordt aangesproken op zijn analytisch vermogen, en dat in een bedrijfskundige context. Maar er is er ook zo een, in een technische context, en in nóg drie contexten. De contexten zijn gekozen langs de lijnen van de 'oude bloedgroepen' in het hbo-ict-onderwijs: bedrijfskundige informatica, informatica, technische informatica, communication and multimediadesign en informatie

dienstverlening en management.
Door alle illustraties uit één en dezelfde context bij elkaar te nemen, ontstaat een compleet beeld van de bestaande opleidingen. Maar het profieldocument biedt uitdrukkelijk ruimte aan opleidingen en studenten om een breder competentieprofiel samen te stellen. Zo kan een bachelor of ICT competenties hebben verworven op het gebied van analyseren en ontwerpen bijvoorbeeld in een bedrijfskundige context, terwijl hij andere competenties heeft verworven in bijvoorbeeld een meer technische omgeving.

Internationaal

Met het oog op de oor het HBO-I verwachte verdere internationalisering van het onderwijs, één van de beoogde gevolgen van de Bologna-afspraken, is in het competentieprofiel van de Bachelor of ICT een internationale paragraaf opgenomen. In deze paragraaf worden de belangrijkste competentieprofielen genoemd en kort gekarakteriseerd. Daarin krijgt ook het HBO-I-profiel een plaats. In afwachting vn ontwikkelingen in de richting van bijv. Europese accreditatie, is eerste voorzichtige poging het 'Nederlandse' profiel in internationaal perspectief te plaatsen.

Toekomst

De generieke beschrijving van bouwstenen van competenties is toekomstvast. Het valt immers niet te verwachten dat een bachelor of ICT in de nabije toekomst niet meer over bijv. analytische competenties zou hoeven te beschikken, of dat er belangrijke competenties in de nu beschreven verzameling zouden ontbreken. In de beschrijving van specifieke beroepssituaties wordt veel meer ontwikkeling verwacht. Reden waarom het HBO-I een webversie van het document in voorbereiding heeft, waarop minimaal eens per jaar onderhoud wordt gepleegd. Illustraties die 'outdated' zijn worden daarin aangepast of verwijderd, ten gunste van actuele beschrijvingen. Het HBO-I zal in de komende maanden een procedure inrichten voor dit onderhoud. De komende tijd zal uitwijzen op welke manier de nieuwe profielbeschrijving kan worden gebruikt. Zeker zal het document niet zorgen voor een verkorting van de sollicitatieprocedure. De titel bachelor of ICT alléén geeft de potentiële werkgever onvoldoende inzicht in wat de net-afgestudeerde kan en kent. Daarvoor zal de bachelor via CV, diplomasupplement en/of portfolio de bewijzen moeten aanleveren van de door hem verworven competenties. Zeker ook zal het nieuwe profiel niet dienen als een blauwdruk voor het samenstellen van leerplannen voor hbo-ict-opleidingen. Gelukkig maar, dat geeft hogescholen en studenten immers de gelegenheid eigen invullingen te ontwerpen. Maar het nieuwe profiel legt wel helder vast wat de minimale kwaliteiten van elke bachelor of ICT zijn. En dat is voor alle belanghebbenden een belangrijke stap.
Het boekje is verkrijgbaar bij het bureau van het HBO-I (info@hbo-i.nl); de tekst is op de websites van het HBO-I (www.hbo-i.nl) en de HBO-raad (www.hbo-raad.nl) te raadplegen.

[1] The Bologna Declaration of 19 June 1999, Joint declaration of the European Ministers of Education, The European Higher Education Area

[2] Beroepsprofiel en Opleidingsprofiel HBO-I, Santema e.a., augustus 2000

Hoe emoticon ben jij?
Op weg naar genderinclusief informaticaonderwijs

Henny van der Lei, Hogeschool van Arnhem en Nijmegen
Gertje Joukes, VHTO

Samenvatting
Het lage aantal meisjes dat kiest voor een ict-opleiding, is al jaren zorgelijk. De talloze initiatieven en activiteiten om de instroom van vrouwen in ict-opleidingen en -beroepen te vergroten hebben nog niet het gewenste resultaat opgeleverd. Henny van der Lei en Gertje Joukes pleiten voor een brede, integrale aanpak, zoals bij de Informatica Communicatie Academie in Arnhem.
Keywords: 'de' student kan een man zijn maar ook een vrouw, brede en integrale aanpak, ketenbenadering, rolmodellen, maatschappelijke context van informatica, oriëntatie op het beroepenveld.

Meer dan ooit is de informaticastudent een man en zijn de vrouwelijke studenten vrijwel 'onzichtbaar'. Hierin schuilt het gevaar dat staf en docenten van informaticaopleidingen alleen nog mannen voor ogen hebben bij innovatie van de onderwijsinhoud, aansluiting op de instroommarkt (voorlichting en werving) en aansluiting op de uitstroommarkt. Er zijn echter veel aangrijpingspunten om het informaticaonderwijs aantrekkelijker te maken voor een bredere doelgroep, waaronder meisjes, maar ook jongens die niet tot de traditionele groep 'informatica-kiezers' horen. Belangrijk is dat het informaticaonderwijs zoveel mogelijk van deze aangrijpingspunten *tegelijk* aanpakt én dat er niet na een of twee jaar wordt geroepen 'dat het alweer niet heeft geholpen'.

Nieuwe voorlichtingscampagne ICA

'Hoe Emoticon Ben Jij' is de titel van de nieuwste voorlichtingscampagne van de Informatica Communicatie Academie (ICA) van de Hogeschool van Arnhem en Nijmegen (HAN). Daarin staan emoticons, die een belangrijke rol spelen bij internetcommunicatie door jongeren, centraal. Een dergelijke manier van communiceren is ook onder meisjes populair en met deze campagne hoopt ICA vooral meisjes aan te spreken. Een eerste onderzoek heeft aangetoond dat de website goed gewaardeerd wordt door meisjes.[i] Hetzelfde geldt voor de serie ansichtkaarten met emoticons die ICA heeft uitgebracht.

De ICA-campagne is gestart in maart 2004 en loopt door in het studiejaar 2004/2005. Doel van de campagne is de instroom te vergroten, met name die van meisjes. De instroomcijfers lopen landelijk al een aantal jaren terug en dat geldt voor de instroom van meisjes sterker dan voor die van jongens. Zo was in 2003 de instroom van meisjes in de hbo-opleiding informatica landelijk maar 4,5%. Die terugloop werd onder andere veroorzaakt door de negatieve verhalen die over de ict-arbeidsmarkt de ronde deden. Het Researchcentrum voor Onderwijs en Arbeidsmarkt voorspelt tussen nu en 2008 echter een groei van 12% ten aanzien van

de informaticaberoepen. Daardoor dreigt opnieuw een oud probleem de kop op te steken, namelijk krapte op de ict-arbeidsmarkt. En dat terwijl de Nederlandse overheid zich ten doel heeft gesteld om in 2007 15% meer instroom in bèta/technische opleidingen te realiseren.

Aantrekkelijke opleidingen

Een voorlichtingscampagne die aansluit bij de interesses van meisjes is een van de instrumenten om de aantrekkingskracht van informatica voor meisjes te vergroten. Goede voorlichting is echter niet voldoende om meer meisjes te doen instromen. Ook de opleidingen zelf moeten meisjes aanspreken. Hetzelfde geldt voor de banen in het werkveld die aansluiten op de opleidingen. Er moeten dus op elk van die terreinen activiteiten worden ontplooid.
In de voorlichting gaat het erom een aantrekkelijk én realistisch beeld te schetsen van de opleidingen en van het werkveld dat daarachter ligt. Veel meisjes kunnen zich niet goed voorstellen wat een informaticaopleiding kan inhouden en nog minder welke beroepen en functies je daarna kunt uitoefenen. Omdat er zo weinig meisjes en vrouwen informatica studeren en in de informatica werken, hebben meisjes in het toeleverend onderwijs ook weinig voorbeelden. Wat helpt, is vrouwelijke studenten en afgestudeerden in te schakelen bij de voorlichting.[ii]

Veel meisjes associëren informatica vooral met techniek en met computers. Wat informatica echter in principe voorheeft op de sector bèta/techniek is dat informatica behalve technische ook veel andere aspecten kent. Het is dan ook van belang om met name aan meisjes duidelijk te maken wat het technische gehalte is van elke informaticaopleiding en elk informaticaberoep en welke rol computers erin spelen. Dit is alleen bedoeld om meer duidelijkheid te scheppen. De boodschap moet zó worden gebracht dat meisjes die iets voelen voor een meer technische informaticaopleiding er niet door ontmoedigd raken en niet krijgen aangepraat dat zo'n opleiding 'dus' niets voor hen is.

Uit onderzoek komt naar voren dat meisjes ten aanzien van inhoud en werkvormen van bèta/technisch en informaticaonderwijs belangrijk vinden.[iii]

- zinvolle contexten, nieuwe dingen leren met de dagelijkse realiteit als uitgangspunt, leren op grond van vragen, cases, problemen: voorbeeld-gestuurd onderwijs, probleemgestuurd onderwijs
- bruikbare (nuttige) en aantrekkelijke (creatieve) producten
- aandacht voor gebruikers/klanten
- aandacht voor ethische en ecologische vraagstukken
- multidisciplinariteit, vakkenintegratie, verbanden leggen
- samenwerken, teamwork.

De laatste jaren zijn er diverse nieuwe opleidingen gestart op het snijvlak van informatica/bèta/techniek en disciplines die traditioneel in de belangstellingssfeer van meisjes liggen, zoals gezondheidszorg, ontwerpen en kunst. Deze opleidingen trekken over het algemeen meer meisjes dan de traditionele bèta-, technische en informaticaopleidingen.
Andere aspecten die bèta/technische opleidingen voor meisjes minder aantrekkelijk maken zijn de vaak ongezellige studieomgeving (inrichting van gebouw, leslokalen en werkruimten), het geringe aantal vrouwelijke docenten en een beperkt beroepsbeeld.

Brede aanpak ICA

Al vanaf de oprichting van de hbo-opleiding (Hogere) Informatica heeft deze opleiding weinig meisjes aangetrokken. Op diverse hogescholen zijn activiteiten ontplooid om daarin verbetering te brengen. Zo werd er op de HAN kritisch gekeken naar de manier van voorlichten, er werden vrouwelijke docenten aangetrokken en er werd nagedacht over de omgeving van de studenten. Het mocht er best gezellig uitzien tussen al die techneuten!
Toen Hogere Informatica op de HAN in 1996 een samenwerkingsverband aanging met de HEAO-opleiding Bedrijfskundige Informatica waren de verwachtingen wat betreft de instroom van meisjes hooggespannen. De samenwerking mondde uit in een gezamenlijke propedeuse. Blijkbaar sloot dat de weg af voor studentes uit de HEAO-propedeuse: de meisjes die anders redelijk vertegenwoordigd waren bij Bedrijfskundige Informatica, meldden zich helaas niet aan voor de gezamenlijke propedeuse. Zo daalde het percentage meisjes dat in de ICA-opleidingen instroomde van 9,5% in 2000 naar 6% in 2003. Landelijk lagen de cijfers nog lager: 5% in 2000 en 4,5% in 2003.

Vier jaar geleden besloot de HAN tot een brede aanpak om de informatica-opleidingen aantrekkelijker te maken voor een grotere doelgroep, met name meisjes. De opleidingen werden geclusterd met 'communicatie' als leidraad. Zo ontstond ICA: de Informatica Communicatie Academie.[iv] ICA heeft gekozen voor een gefaseerde longitudinale, integrale aanpak om de instroom van meisjes en de doorstroom en uitstroom van vrouwelijke studenten te vergroten. Daarbij ondervindt ICA veel steun van het bedrijfsleven.
Specifieke instroomeisen zijn losgelaten, zelfs wiskunde is niet vereist. Iedereen met een havo- of mbo-diploma wordt toegelaten. Inspelen op de interesses van een brede doelgroep uit zich bij ICA ook in de aandacht voor de onderwerpen in de diverse projecten (dicht bij de belevingswereld van studenten, zoals de dierentuin en de gezondheidszorg), de werkvormen, de studieomgeving, de begeleiding en de voorbereiding op de arbeidsmarkt. Het gemeenschappelijke propedeusejaar wordt herontworpen en de inhoud van dit herontwerp wordt voorgelegd aan leerlingen in het voortgezet onderwijs, dus leerlingen uit de instroommarkt. Hun feedback zal worden meegenomen.

ICA heeft inmiddels een nieuwe afstudeerrichting binnen de opleiding Communicatiesystemen: Communicatie & Multi-media Design, op het snijvlak van informatica, communicatie en kunst. De toekenning van de aanvraag vond laat in het schooljaar plaats, zodat er nog weinig bekendheid aan de nieuwe afstudeerrichting kon worden gegeven. Over het effect van deze richting op de instroom van meisjes valt dus nog weinig te melden.
Sinds de oprichting is ICA op zoek naar meer vrouwelijke docenten.

Ketenbenadering

De voorlichtingscampagne 'Hoe Emoticon Ben Jij', waarmee dit artikel begon, staat niet op zichzelf maar is een onderdeel van de ketenbenadering van ICA. Dat begint al op de *basisschool*. Leerlingen worden daar op een alternatieve manier in aanraking gebracht met programmeren. Verder geven ICA-studentes aan meisjes in het basisonderwijs een cursus chatten en leren ze hen hoe ze een digitaal fotoalbum kunnen maken.

De HAN kent *Profielkeuzedagen*, voorafgaand aan de keuze voor een profiel in het voortgezet onderwijs. Deze dagen zijn ontwikkeld op verzoek van en in samenwerking met toeleverende vo-

scholen in de regio Arnhem/Nijmegen. Inmiddels bezoeken per jaar ruim tweeduizend leerlingen uit *3 havo* deze dagen van de HAN. ICA geeft er op een speelse manier een workshop over kenmerken uit het profiel van de informaticus. Kennis over jezelf en communiceren daarover, met andere woorden eerst jezelf kennen voordat je een profiel kunt kiezen, wordt door meisjes erg gewaardeerd. In deze workshop is geen computer te bekennen.
Voor leerlingen uit *4 havo en 4/5 vwo* die al naar Open Dagen en voorlichtingsbijeenkomsten zijn geweest, heeft ICA de LOB-module *Van scriptgirl tot ontwerper* ontwikkeld. Deze module is bedoeld om leerlingen in het voortgezet onderwijs, met name meisjes, te laten kennismaken met de inhoud van de snijvlakopleiding Communicatie & Multimedia Design. De module wordt uitgevoerd door een vrouwelijke docent en bij aanmelding wordt positief gediscrimineerd: meisjes gaan voor. De module leert de deelnemers hoe moderne technologie de communicatie kan ondersteunen. Dat doen ze door een elektronische wenskaart te maken. Eerst ontwerpen ze een script op papier, dat ze vervolgens vertalen in een storyboard. Softwarepakketten voor het bewerken van geluid, foto's en plaatjes helpen hen bij het maken van animaties. De pilotuitvoering heeft inmiddels plaatsgevonden en de waardering is hoog. Het enige 'probleem' is van organisatorische aard: vijf middagen naar ICA komen is voor sommige vo-leerlingen roostertechnisch lastig.

Ook voor de andere ICA-opleidingen zijn dergelijke LOB-modulen in ontwikkeling. Het is de bedoeling dat deze modulen eveneens voor meisjes aantrekkelijk zijn.
Eindexamenkandidaten in het vo krijgen drie aanbiedingen: *workshops, bedrijfspresentaties* (ook voor *mbo*) en ondersteuning bij het *profielwerkstuk*. Het streven was om de inhoud van minimaal twee van de vier workshops te laten bestaan uit oplossingen die je met techniek kunt maken om mensen te helpen en niet uit techniek om de techniek. In 'Maak je eigen videoclip' leren deelnemers een idee te vertalen van woord naar beeld. Dit is echt een thema voor de opleiding Communicatie & Multimedia Design. In de workshop '1, 2, 3, Pizza! Van bestelling tot aflevering' nemen ze het bedrijfsproces van een pizzeria onder de loep. Dit is een workshop in het kader van de ICA-opleiding Bedrijfskundige Informatica. Beide workshops worden uitgevoerd zonder hulp van de computer, behalve dan de laptop en beamer van de docent.

Eindexamenkandidaten uit de regio (vo en mbo) zijn in juni 2004 uitgenodigd voor een bedrijfspresentatie: voor elke opleiding een presentatie door een 'passend' bedrijf. Goede contacten van ICA met het bedrijfsleven maken deze activiteit mogelijk. De insteek is: wat kun je na afronding van de opleiding doen of worden? De rol van communicatie krijgt hierbij veel aandacht. Zelf mbo'ers zijn verrast en enthousiast.
Verder wordt er gewerkt aan de mogelijkheid via Kennisnet om leerlingen uit het vo te ondersteunen bij hun profielwerkstuk.
Sinds 2003/2004 worden *eerstejaars* vrouwelijke ICA-studenten ondersteund door een mentor, een vrouw die al enkele jaren in het beroepenveld werkt en zelf een informaticaopleiding heeft gevolgd. Hieronder zijn ook oud-studenten van de HAN. De resultaten van vorig studiejaar worden momenteel geëvalueerd.

Verdere plannen

Maar daar blijft het niet bij. Zo neemt ICA deel aan Join IT, een project met een vernieuwende aanpak om meer vrouwen te betrekken bij de ict. Join IT, op initiatief

van de VHTO, is een gezamenlijke activiteit van onderwijsinstellingen (vo, hbo, wo), belangenorganisaties, docenten-opleiders, beroepsverenigingen, netwerken, expertisebureaus en bedrijven: een breed samengestelde groep van organisaties die de handen ineen slaan om werk te maken van een grotere participatie van meisjes in de ict.

Een van de directeuren van ICA, Ella Hueting, raakt er steeds meer van overtuigd dat meisjes het beter doen in aparte meisjesklassen. In het verleden waren er heel wat meisjes op aparte meisjesscholen die kozen voor de exacte richting en daar ook uitstekende prestaties behaalden. In Zuid-Europese en Aziatische landen bestaan nog steeds aparte meisjes-scholen, waar meisjes goed presteren, ook in exacte en technische richtingen. In Nederland is sinds het verdwijnen van de meisjesscholen op confessionele grondslag coëducatie de norm. Critici van apart meisjesonderwijs vinden dat dit het 'probleem' alleen maar naar achteren verschuift. Uit recente experimenten in onze contreien, zoals in Zweden, blijkt echter dat de vrouwen die tijdelijk een 'single sex-programma' in Computer Science and Engineering volgden, geen moeilijkheden ondervonden bij de overstap naar het reguliere programma. Ook waarschuwen critici ervoor dat aparte meisjesgroepen voor bèta/technische vakken kunnen worden opgevat als parallellklas voor 'achterstands'leerlingen/ studenten waar een eenvoudiger inhoud wordt gedoceerd. Onderwijs-instellingen met een aparte meisjesgroep om meisjes beter te laten presteren moeten dus gelijke verwachtingen hebben van de mogelijkheden en capaciteiten van jongens en meisjes. Zij moeten dat ook duidelijk naar buiten brengen. Daarom ook heeft Ella Hueting een 'topklas' van meiden voor ogen, met extra begeleiding door docenten en met inhoudelijke en financiële input van bedrijven. Het is een experiment waard!

Om meer zicht te krijgen op de keuzemotieven van meisjes wordt overwogen te onderzoeken waarom meisjes die wel hebben overwogen een informaticaopleiding te kiezen, maar dat uiteindelijk toch niet hebben gedaan. Dit kan aanwijzingen opleveren om de voorlichting, het curriculum, de studie-omgeving en dergelijke opnieuw kritisch onder de loep te nemen.

Besluit

Het zou mooi zijn als we konden besluiten met cijfers die wijzen op een groeiende participatie van meisjes in de ICA-opleidingen. Maar daarvoor is het te vroeg. Bij dergelijke initiatieven in het informaticaonderwijs gaat het om doorgaan en volhouden. Van groot belang is ook samenwerking tussen alle partijen die betrokken zijn bij informatica-onderwijs, zoals in het project Join IT.

Referenties

M. Valkenburg (2001), *Het imago van I*, HBO-I stichting, Amsterdam

Cocky Booy en Gertje Joukes (2004), *Ruim baan voor vrouwelijk talent*, Axis, Delft

VHTO, *Vrouwen in de* ICT*,* Topics, september 2003

I. Kluvers en V. Goedhart (1986), *Hoe apart zijn meisjes? Meisjes apart voor exacte vakken*, Ministerie van OCW, Den Haag

A. van Lenning e.a. (1995), *Inzichten uit Vrouwenstudies; uitdagingen voor beleidsmakers*, Ministerie van SZW, Den Haag

[i] http://www.hoeemoticonbenjij.nl/i_ica.html
[ii] Dat is bijvoorbeeld het geval bij de 'Methode Promoteam' die de VHTO samen met hbo- en mbo-opleidingen heeft ontwikkeld: vrouwelijke studenten spelen een rol bij voorlichtingsactiviteiten zoals open dagen, voorlichtingspraatjes in het toeleverend onderwijs. Daarbij benaderen ze de meisjes actief en staan open voor specifieke informatievragen van meisjes in het toeleverend onderwijs.
[iii] Zie bijvoorbeeld Annita Alting (1987), Boltjes (2004), Van Eck/Volman (1999), Snoeck (2002), Van Aerschot (2003), Booy/Joukes (2004).
[iv] ICA kent momenteel de volgende studieprogramma's die ict en communicatie combineren: Communicatiesystemen/Communicatie & Multimedia Design, Bedrijfskundige Informatica, Informatica en Technische Informatica.

PRO::ICT: voor meer meisjes en vrouwen in de ICT

Eliane Smits, Gertje Joukes, VHTO

Uit diverse Europese onderzoeken blijkt dat het aantal meisjes/vrouwen dat in ICT-opleidingen en beroepen instroomt laag is. Natuurlijk ligt deze problematiek niet in elk Europees land op hetzelfde gebied. In Nederland is een lage instroom van meisjes in ICT-opleidingen te zien in de gehele onderwijskolom (vmbo, hbo, wo). Daarentegen stromen in Groot Brittannië meer meisjes in bij ICT-opleidingen, maar hebben zij na succesvolle afronding problemen om de ICT-arbeidsmarkt te betreden. PRO::ICT wilde deze problematiek verder uitdiepen alvorens met de praktische uitvoering van het project te beginnen.

Uit het inventariserende onderzoek van PRO::ICT bleek wederom dat Nederland op Europees niveau het laagste scoort als het gaat om het aantal meisjes/vrouwen dat een ICT opleiding volgt en vervolgens succesvol afrond. Landen als Italië (37%), Zweden (33%) en Groot Brittannië (33%) kennen een veel hoger percentage vrouwelijk afgestudeerden in techniek/bèta dan in Nederland (17%). Hier zijn verschillende oorzaken voor aan te wijzen.

Door de Onderwijsinspectie (Onderwijsverslag 2002/2003) wordt gesproken van een zogenaamde sorteercultuur op scholen in het voortgezet onderwijs, waar meisjes bij voorbaat al niet aangemoedigd worden voor een natuurprofiel te kiezen. Bovendien, waarom valt het informatica-keuzevak eigenlijk onder het natuurprofiel, als de informatica-vervolgopleidingen in het hoger onderwijs geen natuurprofiel vereisen?

Tevens lijkt het curriculum van opleidingen in informatica en informatiekunde over het algemeen weinig rekening te houden met de vrouwelijke student. De onderwerpen binnen het projectonderwijs worden bijvoorbeeld vaak door een mannelijke docenten bepaald waarbij er in veel gevallen voorbij wordt gegaan aan de belevingswereld van vrouwen. Ook blijken meisjes vooral geïnteresseerd te zijn in de toepassingen van ICT en de wijze waarop die toepassingen worden gecommuniceerd naar de gebruiker. Met dit soort aspecten zou het ICT onderwijs meer rekening kunnen houden. Een kanttekening daarbij is dat de ontwikkeling van de zogenaamde snijvlakopleidingen, waarbij een technische en niet-technische discipline worden gecombineerd, al wat successen hebben geboekt ten aanzien van de instroom van een bredere groep studenten, waaronder meisjes.

Theoretisch kader

Binnen PRO::ICT wordt gekeken naar drie principes: mindset, match and market. Bij de mindset binnen dit project gaat het erom de beeldvorming over ICT van meisjes/vrouwen positief te beïnvloeden. Nog maar al te vaak wordt ICT

geassocieerd met computernerds en saai programmeren. De interactieve kant van ICT, zoals het contact met gebruikers, wordt door veel meisjes/vrouwen over het hoofd gezien. Dit heeft ook met een gebrekkige informatievoorziening te maken omtrent beroepsprofielen.
De match binnen PRO::ICT heeft betrekking op de aansluiting van het onderwijs op de wensen en behoeften van de arbeidsmarkt. Deze tijd vraagt om multidisciplinaire, communicatieve ICT'ers, die zeker dienen te weten hoe je moet programmeren, maar meer in hun mars moeten hebben dan dat. Hier proberen ICT-opleidingen meer op in te springen, maar de geldende eind-competenties staan soms innovaties in de weg.
Tot slot is het van belang dat als vrouwelijk afgestudeerden in ICT op de arbeidsmarkt terecht komen, zij ook op deze markt blijven. Het is om deze reden van belang gunstige randvoorwaarden te creëren die hen behouden voor de ICT branche. Deze randvoorwaarden kunnen bijvoorbeeld flexibele werktijden of kinderopvangregelingen inhouden. Tevens is het van belang dat deze vrouwen een goede loopbaanbegeleiding krijgen en ondersteuning in hun keuzes. Een goed en op diversiteit gespitst personeelsbeleid is hierbij een pré.

Doelstelling
Op grond van het inventariserende onderzoek is een online database ontwikkeld; zie www.pro-ict.net. In deze database zijn modulen met gendersensitief trainingsmateriaal en good practices opgenomen die door de verschillende *directe* doelgroepen (gebruikers) kunnen worden gebruikt, namelijk:

- docenten in het voortgezet onderwijs en het hoger (ICT-)onderwijs
- beroepenvoorlichters
- Human Resource Managers in de ICT-sector

Indirecte doelgroepen (ontvangers) zijn:

- meisjes in het voortgezet onderwijs, voor de keuze van een vervolg-opleiding
- meisjes/vrouwen in hogere ICT-opleidingen
- meisjes/vrouwen die afstuderen (ICT-opleiding) en de arbeidsmarkt betreden.

Meisjes en vrouwen worden dus in *drie* fasen aangesproken, waardoor verschillende acties elkaar kunnen versterken.

Bereik online database
Doelen van de inzet van het materiaal in de database zijn:

- docenten, beroepenvoorlichters en human resource managers materiaal aanreiken dat een positieve beïnvloeding van het beeld van meisjes over technische opleidingen en beroepen/functies kan bewerkstellingen
- docenten, beroepenvoorlichters en human resource managers materiaal aanreiken voor het optimaliseren van de aansluiting tussen de technische opleidingsachtergrond (hoger onderwijs) van meisjes en hun verwachtingen van de (technische) arbeidsmarkt.
- meisjes/vrouwen informatie geven over beroepsmogelijkheden in de ICT-sector/nieuwe media (trends in beroepscompetenties, beroepsprofielen, rolmodellen, etc.).

Binnen diverse workshops en e-learning cursussen voor de directe doelgroepen zal (trainings)materiaal getest worden dat meer meisjes/vrouwen kan betrekken en behouden voor de ICT.

Betekenisvol leren van informatievaardigheden: enkele ontwerprichtlijnen voor instructie

Iwan Wopereis, OTEC/OUNL
Maarten van Veen, OUNL/RdMC

Samenvatting
Leerlingen krijgen meer en meer te maken met activerende didactische werkvormen waarbij zelfstandig of in kleine groepen aan opdrachten wordt gewerkt. Dit vraagt het vermogen om een informatiebehoefte te onderkennen, in vragen te vertalen, informatie op te sporen, te beoordelen, te verwerken en er over te kunnen communiceren: Informatievaardigheden. De Onderwijsraad constateert in het rapport WWW.WEB-LEREN.NL (2003) dat de instructie die gericht is op de verwerving van deze vaardigheden in het huidige onderwijs vaak tekort schiet. De Open Universiteit/ Ruud de Moor Centrum ontwikkelt hulpmiddelen voor docenten in het VO en BVE voor het ontwikkelen van Informatievaardigheden bij leerlingen. Deze bijdrage presenteert een aantal ontwerprichtlijnen voor instructie die docenten en onderwijsontwikkelaars kunnen helpen bij het ontwikkelen of verbeteren van het onderwijs in Informatievaardigheden.

Keywords
Informatievaardigheden, voortgezet onderwijs, instructie, docenten, professionalisering

1. Inleiding

In het onderwijs wordt steeds meer aandacht besteed aan informatievaardigheden (Boekhorst, Kwast, Wevers, 2004). Dat is niet verwonderlijk, aangezien we in een informatiesamenleving leven waarin het vermogen om nieuwe kennis en vaardigheden te verwerven steeds belangrijker wordt. In bijvoorbeeld het Studiehuis van de Tweede fase neemt het bevorderen van een zelfstandige leer- en werkhouding een belangrijke plek in. De wijze waarop we leerlingen en studenten informatievaardig proberen te maken is echter voor verbetering vatbaar. Te vaak beperkt instructie in informatievaardigheden zich tot het aanleren van routinematige technische handelingen die in aparte op zich staande modules wordt aangeboden aan leerling en student. Te weinig worden informatievaardigheden geleerd in een context die gericht is op het betekenisvol leren toepassen van de vaardigheden om problemen op te lossen. Te weinig wordt rekening gehouden met het feit dat we in een dynamisch samenleving leven, waarin instructie in routinematige instrumentele vaardigheden snel verouderd is. Te weinig wordt in het onderwijs de instructie in informatievaardigheden binnen verschillende vakdomeinen met elkaar vergeleken en afgestemd.

In deze bijdrage zoomen we in op een aantal problemen die hierboven worden genoemd. Allereerst wordt kort ingegaan op wat informatievaardigheden zijn. We vervolgen het paper met waarom probleemoplossen centraal moet komen te staan bij onderwijs in informatievaardigheden. Tot slot worden een aantal ontwerprichtlijnen gegeven die docenten en onderwijsontwikkelaars kunnen helpen bij het vormgeven van instructie voor het aanleren van informatievaardigheden.

2. Informatievaardigheden

Informatievaardigheden zijn vaardigheden die worden aangewend om te voorzien in een informatiebehoefte. Deze informatiebehoefte kan worden gezien als een probleem dat opgelost moet worden en daarom wordt in dit kader ook wel gesproken over het oplossen van informatieproblemen (Brand, Wopereis & Vermetten, in druk; Eisenberg & Berkowitz, 1990, 1992; Moore, 1995, 1997). Voorbeelden van informatieproblemen zijn het willen weten hoe laat je met het openbaar vervoer op een bepaalde bestemming komt en het uitvoeren van een uitgebreid literatuuronderzoek naar het effect van Internet op het gedrag van basisschoolkinderen. Informatieproblemen variëren dus in complexiteit: het uitvoeren van een literatuuronderzoek vraagt om meer vaardigheid, dan het zoeken en vinden van de juiste treintijden om op een bepaald station te komen. In het eerste geval dient om het informatieprobleem op te lossen meer en complexere oplosstrategieën toegepast te worden en meer en ingewikkelder tools gehanteerd te worden. Daarnaast vormt bij het uitvoeren van een literatuuronderzoek de regulatie van het proces een grotere component dan bij het zoeken en vinden van feitelijke kennis (zoals treintijden, geboortedata van popsterren, of een pseudoniem van een schrijver).

3. Probleemoplossen

In onze huidige dynamische samenleving is het adequaat kunnen oplossen van problemen van groot belang. Het is daarom niet verwonderlijk de in het onderwijs het leren probleemoplossen een prominente plaats inneemt. Niet alleen de samenleving vraagt in grotere mate om goede probleemoplossers, ook in het onderwijs zelf is men er steeds meer van overtuigd dat wil leren betekenisvol zijn, dit gericht moet zijn op het leren oplossen van authentieke relevante problemen (De Jong & Ferguson-Hessler, 1998; Jonassen, Howland, Moore, & Marra, 2003). Het oplossen van informatieproblemen krijgt in het onderwijs ook steeds meer aandacht. Deels heeft dit te maken met het feit dat we in onze informatiemaatschappij en het onderwijs steeds meer te maken krijgen met het oplossen van ingewikkelde informatieproblemen. Daarnaast is het oplossen van informatieproblemen vaak een belangrijk onderdeel van het oplossen van grotere problemen. Hierbij kun je bijvoorbeeld denken aan het verzamelen van informatie voor het oplossen van een dilemma.

4. Informatievaardigheden en informatieproblemen

Om informatieproblemen op te lossen pas je informatievaardigheden toe. Deze vaardigheden zijn sterk aan elkaar gerelateerd en kunnen worden opgedeeld in vaardigheden die na voldoende oefening routines worden en vaardighedenen die afhankelijk van de situatie telkens op een andere manier worden uitgevoerd (de non-routines). Voor het ontwerpen van onderwijs is het van groot belang om informatieproblemen en benodigde informatievaardigheden in kaart te brengen. Dit in kaart brengen van de vaardigheden die nodig zijn om informatieproblemen op te lossen wordt wel het opstellen van een vaardighedenhiërarchie genoemd (Janssen-Noordman & Van Merriënboer, 2002; Van Merriënboer, 1997). Informatieproblemen oplossen dient in bovenstaand kader te worden opgevat als een complexe cognitieve vaardigheid die bestaat uit een groot aantal aspecten (de informatievaardigheden). Deze informatievaardig-

heden moeten niet in isolatie worden aangeleerd. Het oplossen van een informatieprobleem moet als een geheel worden aangeleerd. Dit betekent voor het onderwijs de toepassing van een 'hele taak benadering', waarin het gehele proces van behoefte aan informatie tot verwerking van de gevonden en geleerde informatie aan bod komt.

5. Richtlijnen voor onderwijs in informatievaardigheden

Goede instructie voor het (aan)leren van complexe vaardigheden, zoals het oplossen van informatieproblemen, vergt een gedegen aanpak. Er zijn handboeken die het ontwerpproces voor goede en effectieve instructie voor complexe vaardigheden uitvoerig beschrijven (zie bijvoorbeeld Van Merriënboer, 1997). In deze bijdrage kunnen en willen we niet een gehele ontwerpmethodiek beschrijven. Hiervoor verwijzen we graag naar Janssen-Noordman en Van Merriënboer (2002). Een aantal aspecten bij het ontwikkelen van goede instructie willen we echter in deze paragraaf benadrukken en aan de orde stellen.

- Ga uit van een hele taak benadering. Informatievaardigheden, zoals het hanteren van een zoekmachine en het opstellen van een zoekvraag, dienen niet geïsoleerd aangeboden te worden. Uiteindelijk gaat het bij informatie-vaardigheden om het kunnen toepassen ervan in een betekenisvolle context. Dit betekent dat voor goede instructie realistische informatieproblemen verzameld dienen te worden die het uitgangspunt moeten vormen van de constructie van leertaken. Leertaken, waarin het gehele proces van informatieprobleem oplossen aan bod komt, moeten het uitgangspunt zijn van de instructie.
- Rangschik leertaken van eenvoudig naar complex. De leertaken/ informatieproblemen die worden verzameld dienen te worden gerangschikt van eenvoudig naar complex. Biedt deze taken ook van eenvoudig naar complex aan. Voor zowel eenvoudige als complexe taken geldt dat het uitgangspunt moet zijn dat het gehele oplosproces doorlopen wordt.
- Laat de ondersteuning afnemen. Binnen een vakdomein en liever over vakdomeinen heen dienen informatie-problemen verzameld te worden en geclassificeerd naar complexiteit (zie vorige richtlijn). Taken van dezelfde complexiteit dienen vervolgens in zogenaamde taakklassen ondergebracht te worden. Binnen een taakklasse laat je vervolgens de ondersteuning van de docent of het materiaal afnemen. Het principe is dat leerlingen binnen een taakklasse steeds meer zelfstandig een informatieprobleem van dezelfde complexiteit leert op te lossen. Binnen een taakklasse zou je bijvoorbeeld allereerst een uitgewerkt voorbeeld of modelling example kunnen presten-teren aan de leerling, vervolgens een taak waarin een deel van de oplossing reeds is gegeven en tot slot een conventioneel probleem, waarin de leerling het probleem zelfstandig oplost (zie voor meer toelichting op taakklassen en type taken Janssen-Noordman & Van Merriënboer, 2002).
- Bied leertaken met informatie-problemen uit verschillende contexten aan. Zorg ervoor dat de context waarbinnen een informatieprobleem speelt varieert. Dit is belangrijk voor transfer van leren (Perkins & Salomon, 1989). Met transfer wordt bedoeld dat leerlingen het geleerde in nieuwe,

onbekende situaties moet kunnen gebruiken. Door een grote variatie aan problemen en probleemcontexten aan te bieden, bouwen de leerlingen aan een rijk repertoir van oplosmogelijkheden.

- Stem instructie in het oplossen van informatieproblemen binnen verschillende vakdomeinen (vakgebieden) op elkaar af. Het oplossen van informatieproblemen komt in alle vakgebieden voor. We zagen eerder dat het belangrijk is dat leerlingen informatieproblemen in verschillende contexten (bijvoorbeeld vakgebieden) aanleren. Door informatieproblemen binnen verschillende vakgebieden met elkaar te vergelijken, kan wellicht gekomen worden tot afspraken die ertoe leiden dat bepaalde routinematige en niet-routinematige aspecten ('deelvaardigheden') meerdere malen tijdens de opleiding worden aangeboden. Door goed af te stemmen kunnen taakklassen met leertaken die gericht zijn op het oplossen van informatieproblemen een domeinoverstijgend karakter krijgen.
- Besteed aandacht aan de regulatie van het oplosproces. Het oplossen van informatieproblemen is een proces van plannen en uitvoeren van zoekstrategieën, evalueren van (tussen) resultaten en het verifiëren of informatie juist is (triangulation). Afhankelijk van de opbrengsten van de verschillende activiteiten in het oplosproces ga je verder of terug in het proces. Het proces is dus verre van lineair. Het plannen, monitoren en evalueren van het proces is daarom erg belangrijk en binnen de instructie dient hier veel aandacht aan besteed te worden. Dit kan gebeuren door gebruik te maken van een 'cognitive apprenticeship' benadering (Collins, Brown, & Newman, 1989). Bij deze benadering leren leerlingen het probleemoplosproces te reguleren door een combinatie van observatie en geleide instructie. De docent modelleert, coacht en laat de instructie afnemen. Bij het informatieproblemen oplossen, kun je per taakklasse een modelling example toevoegen, waarbij de expert (de docent) het gehele proces illustreert en die hardop denkend doet. Hierbij kan de docent accenten leggen op de keuzes die hij/zij maakt en extra aandacht besteden aan het plannen, monitoren en evalueren van de (leer)taak.

Referenties

- Boekhorst, A., Kwast, I., & Wevers, D. (2004). *Informatievaardigheden* (3e druk). Utrecht: Lemma.
- Brand-Gruwel, S., Wopereis, I., & Vermetten, Y. (in press). Information problem solving: Analysis of a complex cognitive skill. *Computers in Human Behaviour*.
- Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the draft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction. Essays in honor of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum.
- De Jong, T, & Ferguson-Hessler, M. G. M. (1998). Leren probleemoplossen. In L. Verschaffel & J. Vermunt (Eds.), *Het leren van leerlingen [Onderwijskundig Lexicon editie III] (pp. 65-80)*. Alphen aan den Rijn: Samsom
- Eisenberg, M. B., & Berkowitz, R. E. (1990). *Information problem-solving: The Big Six Skills approach to library*

and information skills instruction. Norwood, NJ: Ablex.
- Eisenberg, M. B., & Berkowitz, R. E. (1992). Information problem-solving: The big six skills approach. *School Library Media Activities Monthly*, *8*(5), 27-29, 37, 42.
- Janssen-Noordman, A. M. B., & Van Merriënboer, J. J. G. (2002). *Innovatief onderwijs ontwerpen: Via leertaken naar complexe vaardigheden*. Groningen: Wolters-Noordhoff.
- Jonassen, D., Howland, J., Moore, J., & Marra, R. (2003). *Learning to solve problems with technology: A constructivist perspective*. Upper Saddle River, NJ: Prentice Hall.

Moore, P. (1995). Information problem solving: A wider view of library skills. *Contemporary Educational Psychology*, *20*, 1-31.
- Moore, P. (1997). *Teaching information problem solving in primary schools: An information literacy survey*. Paper presented at the 63rd IFLE general conference, Copenhagen Denmark.
- Onderwijsraad (2003). *WWW.WEB-LEREN.NL*. Den Haag: Auteur.
- Perkins, D. N., & Salomon, G. (1989). Are cognitive skills context-bound? *Educational Researcher*, *18*, 16-25
- Van Merriënboer, J. J. G. (1997). *Training complex cognitive skills*. Englewood Cliffs, NJ.: Educational Technology.

Eerstegraads lerarenopleiding informatica

Bert Zwaneveld, Open Universiteit
Betsy van Dijk, Universiteit van Twente

CODI, bedoeld om met de invoering van informatica als keuzevak in de bovenbouw van havo en vwo voor bevoegde eerstegraads docenten te zorgen, is aan het eind van zijn looptijd. Het is nu dus het moment om tot een reguliere eerstegraads lerarenopleiding informatica komen.

Sinds ongeveer twee jaar is op initiatief van CODI een aantal mensen bezig de opzet en inhoud van die eerstegraads lerarenopleiding uit te denken. Dat werk is nu klaar. Zes universiteiten, die van Groningen, Utrecht, Nijmegen, Twente, Eindhoven en Delft (de laatste drie samenwerkend als de TULO, de Technische Universitaire Leraren Opleiding) willen graag starten. Hun bedoeling is om de lerarenopleiding informatica als een nieuwe 'track' naast hun andere bèta-tracks in te voeren. Inmiddels heeft het ministerie van OCW laten weten het goed doorstaan van de toetsen, 'toets nieuwe opleiding' door de Nederlands-Vlaamse Accreditatieorganisatie en de doelmatigheidstoets door het ministerie, als voorwaarde voor erkenning van de komende, zogeheten educatieve master in informatica te vergen.

Recent hebben vertegenwoordigers van de genoemde instellingen besloten om de gang langs de accreditatieorganisatie en het ministerie te gaan maken. Zij zullen daarbij optimaal van elkaars en in CODI-verband opgedane expertise gebruik te maken. Als alles volgens plan verloopt kan dan op 1 september 2005 de reguliere eerstegraads lerarenopleiding informatica van start gaan.

Het ontwikkelde opleidingsmodel mikt op excellente informatici en excellente leraren. Zonder hier op alle details in te gaan ziet de beoogde educatieve master in informatica er als volgt uit. De opleiding duurt twee jaar en is bedoeld voor mensen die bachelor (of science) in informatica zijn, dus zowel universitaire als hbo-bachelors zijn toelaatbaar. In de tweejarige masteropleiding gaat het om het verwerven van competenties op de volgende gebieden:

- het beroep van leraar
- de vakinhoud
- de vakdidactiek
- het schoolvak informatica en de ontwikkeling ervan
- persoonlijke competenties.

Over het geheel van de bachelor- plus masteropleiding gaat het om de volgende vakinhoudelijke competenties:

- software: programma's en algoritmiek
- gegevens: informatie- en kennissystemen
- hardware: machines en infrastructuur
- grondslagen
- (research en) development
- omgeving
- informatica als wetenschapsgebied.

Aan de vakdidactische kant gaat het om:

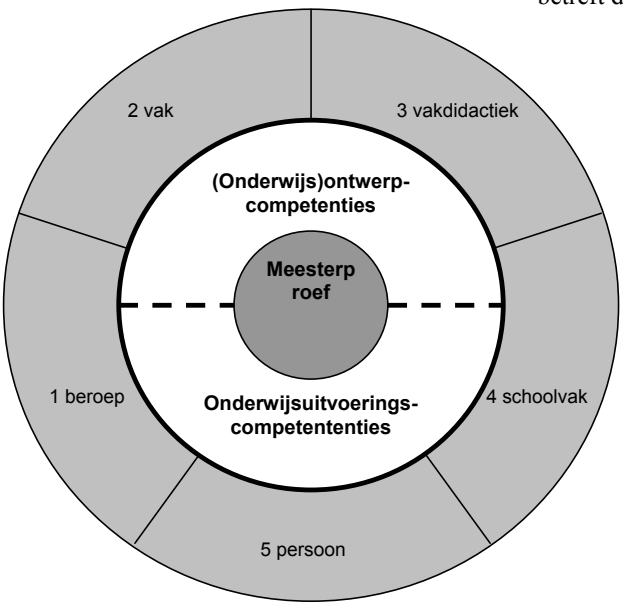
- vakdidactiek
- algemene didactiek
- schoolpracticum
- informatica-vakinhoud in relatie tot het schoolvak en de ontwikkeling ervan.

Bij de vakdidactiek en de schoolvakontwikkeling gaat het onder andere om de volgende thema:s: didactiek van programmeren, van informatiesysteemontwikkeling, van gegevensbanken en SQL; het werken met websites als rode draad voor het praktische werk; leerstofordening rond informaticabegrippen; samenwerkend leren, groepswerk, zelfstandig leren; projectwerk en eventueel profielwerkstuk; computerpractica; demonstraties (vooral ook door leerlingen); visieontwikkeling, bijvoorbeeld op de verhouding tussen het alfa-, bèta- en gammakarakter van het vak; wat zijn de 'relatief constante' kernbegrippen en kernvaardigheden, en welke nieuwe ontwikkelingen moeten in het curriculum opgenomen worden; samenwerking met de andere schoolvakken; samenwerking met collega informaticadocenten van andere scholen voor voortgezet onderwijs (voor informatica als keuzevak zal er per school immers meestal één leraar zijn, zodat er geen vaksectie informatica per school is, wellicht wel per groep van scholen).

Onderstaande figuur brengt het opleidings ontwerp in beeld.

De *kern* van de opleiding is de meesterproef: het leveren van een substantiële bijdrage aan de ontwikkeling van het schoolvak of de vakdidactiek. Dit betreft het ontwerpen, implementeren, evalueren en aanpassen van informaticaonderwijs. Door het uitvoeren van deze opdracht laat de aanstaande eerstegraads leraar informatica zien inderdaad competent te zijn voor het uitoefenen van zijn toekomstige beroep.

Vóór het uitvoeren van deze opdracht moet de aanstaande eerstegraads leraar informatica hebben laten zien dat hij beschikt over: voldoende ontwerp-/onderzoekscompetenties en over voldoende onderwijsuitvoeringscompetenties. Dit betreft de *binnenste schil*.



De *buitenste schil* is bedoeld om de leraar in opleiding waar nodig de competenties te laten verwerven die vereist zijn om de competenties van de binnenste schil en de kern met succes te verwerven.

Een opstap naar abstract denken

Elise Boltjes, Noordelijke Hogeschool Leeuwarden

Er zijn weinig meisjes die exacte vakken kiezen. Dat ligt volgens mij niet aan de meisjes, maar aan de gangbare manier van leren en lesgeven. Dit onderzoek herformuleert een informatieanalysemethode tot een leer- en lesmethode die goed aansluit bij de manier waarop meisjes leren. De methode gaat uit van voorbeelden, vandaar dat zij voorbeeldgestuurd onderwijs is genoemd.

Het verschil in schoolprestaties tussen meisjes en jongens blijkt niet te ontstaan door een verschil in cognitief vermogen, maar door een verschil in zelfbeeld ten opzichte van hun score. Meisjes hebben een lager zelfbeeld van hun eigen kunnen dan jongens. Voorbeeldgestuurd onderwijs komt geloofwaardiger over en biedt meer zekerheid, omdat het uitgaat van de onzekerheid van de leerling. Daar voelen meisjes zich beter bij. En jongens ook.

In het onderwijs is bij het “nieuwe leren” een overgang merkbaar van “de leraar vertelt” naar “de leerling vraagt”. Voorbeelden van vormen van het nieuwe leren zijn projectonderwijs, probleemgestuurd onderwijs en competentiegericht onderwijs. Deze vormen van onderwijs zijn niet in klassikaal gebonden onderwijs te gebruiken. Voorbeeldgestuurd onderwijs is echter wel klassikaal toepasbaar en benut tevens de voordelen van het nieuwe leren. Daarbij is een overgang merkbaar van “lesgeven vanuit de zekerheid van de leraar” naar “lesgeven vanuit de onzekerheid van de leerling”.

Een aantal vragen met korte antwoorden:

? Wat heeft je aangezet tot dit onderzoek?

Dat weinig meisjes een technisch profiel kiezen.

? Wat is het verschil in leren tussen meisjes en jongens?

Niet cognitief, maar de mate van zelfbeeld. Als meisjes een 7 halen voor wiskunde denken ze "ik begrijp het eigenlijk niet" en jongens vinden dat ze een wiskundeknobbel hebben. Bij dezelfde 7.

?Hoe heb je dat opgelost?

Meisjes geven meer toe aan hun onzekerheid. Voorbeeldgestuurd onderwijs gaat daarom uit van voorbeelden uit het dagelijks leven die meisjes begrepen hebben. "O, zit dat zo. Dat begrijp ik wel." Voorbeeldgestuurd onderwijs wil leerlingen geen zekerheid bieden, maar de onzekerheid laten accepteren. Je ervaart de grootst mogelijke zekerheid, door het accepteren van je onzekerheid.

? Wat is het resultaat van het onderzoek?

Traditioneel gegeven lessen beoordelen meisjes lager dan jongens. Voorbeeldgestuurd gegeven lessen beoordelen meisjes veel hoger. Jongens beoordelen voorbeeldgestuurd gegeven lessen ook hoger, maar meisjes gaan meer vooruit dan jongens. De beoordeling van beide stijgt tot een gelijkwaardig niveau, zodat het oorspronkelijke verschil verdwenen is.

Iedereen gaat dus vooruit bij gebruik van voorbeeldgestuurd onderwijs.

? Heb je daar een verklaring voor?
Als je uitgaat van meervoudige intelligenties, dan gebruikt het traditionele onderwijs bij exacte vakken vooral de cognitieve intelligenties. Voorbeeld-gestuurd onderwijs benut veel meer de emotionele en sociale intelligentie. Je doet jezelf te kort als je die intelligentie niet optimaal benut. Meisjes kunnen niet zonder, en jongens met hun aangeleerde bravoure ook niet.
Voor inlichtingen zie
www.VoorbeeldgestuurdOnderwijs.nl

Referenties:
Boltjes, E.G. (2004). *Voorbeeld$_{IG}$ Onderwijs. Voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes.* Proefschrift Universiteit Maastricht:
www.VoorbeeldgestuurdOnderwijs.nl.

Promoteam Informatica

Rijks*universiteit* Groningen

Informatiefolder

Wie zijn wij?

Het Promoteam Informatica bestaat uit vijf enthousiaste studenten die scholen bezoeken om bovenbouw VWO leerlingen te informeren over de studie Informatica.

Sylvia

Martijn

Wat doen wij?

Wij verzorgen in een Wiskunde of Informatica lesuur een interessante interactieve les over een Informatica onderwerp, of geven een voorlichting over de studie Informatica met aansluitend een leuk praktisch gedeelte (2 lesuren).

Laurens

Maarten

Contactinformatie

Kijk voor alle informatie op onze website:
http://www.rug.nl/informatica/infpromo
Voor vragen en afspraken kunt u mailen naar:
infpromo@wing.rug.nl of bel Martijn Wieling:
06-42733867. **We komen graag bij u langs!**

Timo

Grafische ondersteuning bij programmeeronderwijs

Jan Kuper, Universiteit Twente

Deze bijdrage beschrijft de ervaringen met een softwarepakket waarmee grafische ondersteuning wordt geboden bij het programmeeronderwijs. De software richt zich op het weergeven van grafische datastructuren (bomen, grafen), maar de positieve ervaringen ermee hebben geleid tot het ontwikkelen van grafische ondersteuning bij andere thema's, zoals wiskundige grafieken, natuurkundige simulatie, computational geometry.
Omdat het pakket is geschreven in een functionele taal, is het op dit moment alleen bruikbaar bij het onderwijs in functioneel programmeren. De benadering kan echter voor het onderwijs in elk programmeerparadigma zinvol zijn.

Boomstructuren

In het programmeeronderwijs worden, ongeacht het gekozen programmeer paradigma, boomstructuren door studenten in eerste instantie veelal als moeilijk ervaren. Studenten hebben vaak relatief veel tijd nodig de practicumopdrachten over dit onderwerp af te ronden waardoor er weinig opdrachten binnen één practicumbijeenkomst aan de orde gesteld kunnen worden. Het verband tussen de syntactische formulering van bomen, en de grafische intuïtie bleek voor veel studenten moeilijk te zijn. Om die reden is een programma ontwikkeld waarmee algemene boomstructuren eenvoudig kunnen worden afgebeeld op het scherm zodat studenten het resultaat van hun programma's zichtbaar kunnen maken. De hoop was dat daarmee een groter aantal opdrachten binnen kortere tijd kon worden gemaakt, waardoor het begrip bij studenten zich beter en sneller zou ontwikkelen. Die hoop is boven verwachting uitgekomen, het aantal opdrachten dat tijdens de eerste practicumbijeenkomsten over bomen wordt afgerond, is vele malen groter dan in het verleden, en bovendien is de moeilijkheidsgraad groter.

Het ondersteunende programma gaat uit van het volgende algemene type (*rosetree* genaamd):

```
roseTree
  ::= RoseNode
        [char]
        [roseTree]
```

Dat wil zeggen dat aan elke knoop (*node*) in een boom van dit type een string kan worden opgeslagen, en bovendien heeft elke knoop een willekeurig aantal subbomen. Een node is een *blad*, als hij geen subbomen heeft. Het type `roseTree` is dus vrij algemeen, en kan diverse structuren weergeven. Een beperking is het feit dat aan de knopen slechts een enkele string kan worden opgeslagen, maar omdat veel waarden (getallen, getallenparen, waarheidswaarden, etc) in stringnotatie kunnen worden weergegeven, wordt deze beperking pas relevant bij het weergeven van gestructureerde en/of multimedia waarden.

Om het programma te kunnen gebruiken, hebben studenten de beschikking over twee functies: `showTree` en `showTreeList` die als argument respectievelijk een enkele boom, of een lijst van bomen meekrijgen. Deze functies kunnen echter alléén op bomen van het type `roseTree` worden toegepast. Om die reden moeten hun eigen boomstructuren eerst omgezet worden naar bomen van het type `roseTree`. Didactisch gesproken blijkt dat echter veeleer een voordeel dan een nadeel te zijn: dergelijke omzet programmaatjes zijn eenvoudige eerste introducties in het schrijven van recursieve boomprogramma's. Als eerste opdracht moeten studenten een binaire bomen definiëren dat getallen aan de bladeren en aan de interne knopen kan bevatten. Een typedefinitie waarmee dergelijke bomen weergegeven kunnen worden is bijvoorbeeld:

```
Boom
  ::= Blad num
    | Knoop num boom boom
```

Studenten moeten vervolgens enkele voorbeeldexpressies van dit type geven, zoals:

```
t1 = Blad 37
t2 = Knoop 15 (Blad 20)
              (Blad 30)
t3 = Knoop 15 t1 t2
```

Boomexpressies lopen snel uit de hand en zijn dan volstrekt onleesbaar voor een menselijke lezer, maar de functie `zetom` om zo'n expressie om te zetten naar het type `roseTree` zodat hij grafisch kan worden getoond, is een zeer eenvoudig eerste voorbeeld van een recursieve functie op boomstructuren:

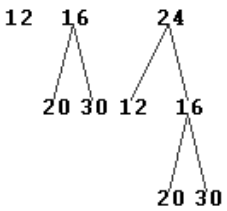
```
zetom (Blad n)
  = RoseNode (itoa n) []

zetom (Knoop n t1 t2)
  = RoseNode (itoa n)
      [zetom t1, zetom t2]
```

De aanroep

```
showTreeList
  (map zetom [t1, t2, t3 ])
```

levert de volgende afbeelding van de drie bomen naast elkaar:



Ditzelfde moeten de studenten doen met enkele varianten van boomtypes, zodat steeds opnieuw een zetom-functie geschreven moet worden. Daardoor wordt het patroon van recursieve functiedefinities op bomen heel snel herkend – en bovendien krijgen studenten snel behoefte aan een algemenere definitie van een zetom-functie, die alle verschillende varianten in één keer definieert, een goede gelegenheid om *generic funcitions* te intorduceren.

Door deze grafische weergave van bomen, kunnen studenten zich concentreren op het schrijven van typische boomfuncties zoals een functie om bomen te spiegelen, bewerkingen op de elementen in een boom uit te voeren, subbomen te selecteren, bomen sorteren, balanceren, etcetera. Het resultaat van hun functies op een voorbeeldboom `b`

kunnen zij onmiddellijk controleren door aanroepen als:

```
showTreeList
  [ b , zetom b
  , zetom (spiegel b) ]
```

Binnen een enkele practicum-bijeenkomst kunnen 15-20 van dergelijke opgaaven worden gemaakt, zodat na afloop studenten blindelings het recursieve schema kunnen produceren.

Grafen

Geïnspireerd door het succes van de grafische ondersteuning bij bomen is er vervolgens hetzelfde gedaan bij opgaven over *grafen*. Bij grafen echter, is geen recursieve definitie van de datastructuur voorhanden, en bovendien is het niet zo duidelijk hoe een graaf middels een expressie kan worden weergegeven. Daarom is een programma ontwikkeld waarmee studenten door muisklikken een graaf kunnen tekenen. Daarnaast hebben zij functies tot hun beschikking waarmee onderdelen van de graaf een andere kleur gegeven kunnen worden. Bij dit onderdeel komen *algebraïsche datatypes* op een zeer natuurlijke wijze naar voren omdat in de gebruikte programmeertaal (*Amanda*, ontwikkeld door Dick Bruin) grafische mogelijkheden daarmee zijn vormgegeven. Voorbeelden van algebraïsche constructoren voor de grafische toepassing zijn `GraphPolyLine`, waaraan een eental punten (gegeven als coördinatenparen) kunnen worden meegegeven die door deze constructor worden verbonden, en `GraphEllipse`, waarmee een ellips getekend kan worden, gegeven twee hoekpunten van de rechthoek die de ellips insluit.

Voor *inputevents* geldt hetzelfde: toetsaanslagen en muisklikken worden ook met waarden in een algebraïsch type weergegeven, bijvoorbeeld `KeyIn`, `MouseDown`.

Met behulp van dergelijke constructoren zijn de functies `drawNode`, `drawEdge`, `drawGraph` geschreven die aan studenten beschikbaar worden gesteld. Deze functies kunnen de punten van een graaf (*nodes*) in verschillende kleuren weergeven, en de bindingen tussen nodes (*edges*) in verschillende diktes en kleuren. Daarmee kunnen de resultaten van graafprogramma's zichtbaar gemaakt worden, zoals het bepalen van de buren van een node, het één voor één weergeven van alle paden van een gegeven node naar een andere node, het geven van dezelfde kleur aan alle nodes van een samenhangende subgraaf, etc.
Behalve het weergeven van het resultaat van een berekening, kan ook het *proces* van die berekening tot op zekere hoogte zichtbaar worden gemaakt. Een aansprekend voorbeeld hiervan is het bepalen van alle paden in een graaf van een gegeven node naar een andere node.

Wiskundig wordt een graaf gedefinieerd als een verzameling nodes samen met een verzameling geordende paren van die nodes. Als we de nodes omwille van de eenvoud als getallen representeren, is een mogelijke definitie van het type van grafen:

```
graph
  ::= {nodes::[num]
      ,edges::[(num,num)]
      }
```

Neem aan dat de functie `buren`, die de lijst van direct aangrenzende buren van een gegeven node in een graaf oplevert, al bestaat. Dan kunnen de paden van een gegeven node `p` naar een node `q` recursief worden bepaald door vanuit alle directe buren van `p` de paden naar `q` te bepalen, en de eerste stap vanuit `p` er voor te "plakken". De recursieve clause van de definitie van de functie `paden` kan dus als volgt geschreven worden:

```
paden g p q
 = concat
    [ map ((p,x):) ps
      | x <- buren p
      ; ps := paden g' x q
      ]
```

Hierin is `g` de graaf, en `p`, `q` zijn de nodes in `g` die door paden verbonden moeten worden. De node `x` varieert over alle directe buren van `p`, en `ps` is de lijst van alle paden van `x` naar `q` binnen de graaf `g'`. Deze is uit `g` ontstaan door de node `p` te verwijderen zodat cykels worden voorkomen. De edge `(p,x)` wordt met de standaardfunctie `map` voor elk pad in `ps` "geplakt".

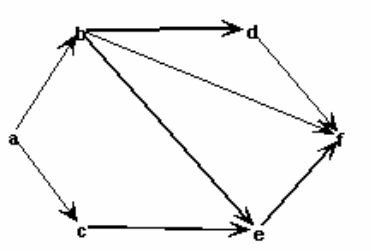
Uiteraard moeten de basisgevallen nog aan de definitie van `paden` worden toegevoegd. Daarin zijn twee gevallen te onderscheiden:

- als we in `q` "aangekomen" zijn, dwz als `p=q`: in dat geval is het resultaat de lijst met alleen het lege pad `[[]]`,
- als we vanuit `p` binnen `g'` niet "verder kunnen", dus als `p` geen buren heeft: in dat geval is het resultaat de lege lijst `[]`.

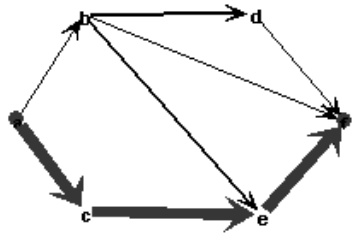
Een pad `pq` is nu een lijst van edges, en die kan visueel worden weergegeven door de functie `drawEdge` met de adequate parameters (bijvoorbeeld 3 voor de dikte van de edge) op elke edge in het pad toe te passen:

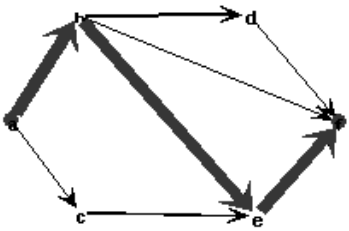
```
map (drawEdge 3 red) pq
```

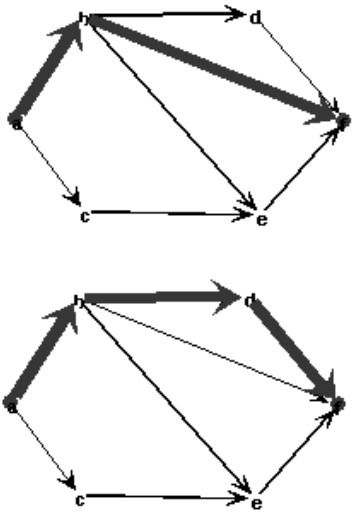
Als we dit algoritme testen op de volgende graaf:



en de paden van `a` (links) naar `f` (rechts) tonen, zien we het volgende resultaat:







Overige toepassingen

Vanwege de uiterst positieve ervaringen met deze ondersteunende methodes, en de indruk dat studenten een grafische output als stimulerend ervaren, is er ook moeite gedaan om andere opdrachten zodanig te kiezen dat een grafische weergave van de resultaten mogelijk is. Die opdrachten zijn gevonden op diverse gebieden:

- *Wiskundige grafieken*: een eenvoudige toepassing van de grafische mogelijkheden is om een functie te schrijven die, gegeven een wiskundige functie en een interval op de *x*-as de grafiek van die functie op dat interval tekent,
- *Computational geometry*: bepaal de convexe hull van een aantal punten in het platte vlak. Die punten moeten door muisklikken worden aangebracht, en de convexe hull moet getekend worden. Andere voorbeelden van opdrachten zijn het bepalen van alle triangulaties op een verzameling punten, het bepalen van een Delauney triangulatie op een verzameling punten, etc. Een bijkomend voordeel van dergelijke opdrachten is het feit dat studenten geconfronteerd worden met afrondingsfouten bij binaire getalrepresentaties. Dat geeft onverwachte effecten die in de grafische representatie nadrukkelijk zichtbaar zijn.
- *Natuurkundige simulatie*: het weergeven van bewegende objecten is voor studenten zeer stimulerend. Voorbeelden van eenvoudige opdrachten zijn stuiterende balletjes, verende elastiekjes, etc. Daarvoor moeten studenten overigens wel enig begrip hebben van de wiskundige beschrijving van bewegeingen. Bij stuiterende balletjes en verende elastiekjes gaat dat niet evrder dan een polynoom, maar mij een slingerbeweging wordt dat ingewikkelder. Een bijzonder aardig voorbeeld is het laten bewegen van een balletje tussen objecten waar het tussen heen en weer stuitert, met name als die objecten verschillende vormen kunnen aannemen,
- *Bord- en kaartspelen*: het achtkoninginnenprobleem is overbekend. Maar om de stukken daadwerkelijk op een schaakbord af te laten beelden geeft een extra dimensie aan de opgave. Het aantal variaties op opdrachten in de sfeer van dergelijke spelletjes is eindeloos. Studenten ervaren opdrachten van deze aard altijd weer als leuk.

Evaluatie

Er is geen systematische evaluatie gedaan van de verschillen tussen een practicum met bovenbeschreven middelen, maar er bestaat een sterke tot zeer sterke indruk dat er een aantal positieve kanten aan zitten. Die indruk is gebaseerd op de volgende observaties:

- student-assistenten kunnen vergelijken met de wijze waarop zij zelf het

betreffende onderwijs hebben ontvangen. Hun reacties waren zeer positief: “leuk, ik wou dat ik zelf het vak op deze wijze had gehad”,
- het is zichtbaar dat de directe feedback die met behulp van de grafische weergave bijvoorbeeld bij boom-programma’s wordt bereikt, een positief effect heeft op de begripsontwikkeling van de studenten,
- in de practicumzaal is zichtbaar dat studenten gemotiveerd en enthousiast met de opdrachten bezig zijn. Ze roepen elkaar: “kom eens kijken!”, en geven elkaar complimentjes over de resultaten. Daarvoor is vermoedelijk bevorderlijk dat niet iedereen dezelfde opdracht zit te maken, maar dat er enige variatie wordt aangebracht. Die variatie is eenvoudig aan te brengen, en bovendien biedt een grafische omgeving de mogelijkheid tot variatie in vormgeving door de studenten zelf. Daar maken ze uitvoerig gebruik van,
- ieder vak aan de opleiding Informatica in Enschede wordt geëvalueerd. Dit vak wordt in die evaluaties als zeer positief beoordeeld. Onder andere wordt veelvuldig het practicum genoemd als een zeer leerzame en stimulerende omgeving.
- bovenstaande observaties zijn gemaakt aan de hand van verschillende studentengroepen: eerstejaars informatica en eerstejaars bedrijfs-kundige informatica studenten, derde jaars wiskunde studenten, derdejaars informatica studenten.

Conclusie

De conclusie is dat grafische ondersteuning bij programmeer-onderwijs een gunstig effect heeft op de motivatie en op het leereffect bij studenten.

Functioneel Programmeren met Helium

Bastiaan Heeren en Daan Leijen
Instituut voor Informatica en Informatiekunde, Universiteit Utrecht

Moderne functionele talen zijn mathematisch precies, notationeel elegant, en sterk getypeerd. Deze talen zijn daarom bij uitstek geschikt voor het onderwijzen van programmeerconcepten en algoritmiek. Helium is een onderzoeksproject waarbij een krachtig typesysteem gecombineerd wordt met precieze en gebruiksvriendelijke foutmeldingen.

Keywords: *puur functioneel programmeren, gebruiksvriendelijke foutmeldingen.*

1 Introductie

Pure functionele programmeertalen zijn uitermate geschikt voor het ontwikkelen van correcte software. De eigenschappen die dit mogelijk maken zijn niet alleen van belang bij software ontwikkeling, maar maken deze talen juist ook interessant voor onderwijs doeleinden.

Imperatieve programmeertalen, zoals bijvoorbeeld Java, C, en C^{++}, beschrijven programma's aan de hand van een serie opdrachten. Deze talen genieten een grote populariteit en worden veelvuldig gebruikt en onderwezen. Pure functionele programmeertalen vormen een alternatief paradigma, waarin programma's worden geschreven als een verzameling van functies zoals bekend uit de wiskunde. Dit vereist een andere manier van programmeren, zonder aandacht te schenken aan bijvoorbeeld berekeningsvolgorde of geheugenallocatie. Recursie, polymorfie, en hogere-orde functies spelen binnen deze talen een centrale rol. Daarnaast lenen zij zich uitstekend om eigen datastructuren te definiëren, en om eigenschappen over programma's te bewijzen (bijvoorbeeld met inductie).

Haskell is zo'n functionele programmeertaal (andere voorbeelden zijn ML en Clean). Haskell is een sterk getypeerde taal: typeringsfouten worden al tijdens compilatie gemeld aan de gebruiker. Deze foutmeldingen zijn echter dikwijls van een cryptische aard voor beginnende programmeurs.

Onze ervaringen met het onderwijzen van Haskell waren destijds de aanleiding voor het ontwikkelen van een betere compiler waarmee studenten sneller en met meer plezier functioneel programmeren kunnen leren. Het resultaat is de Helium compiler, en deze heeft de volgende kenmerken:

- Helium bevat een nieuw algoritme voor typeinferentie. Dit algoritme gebruikt een globale constraint-analyse op een typeringsgraaf om fouten preciezer te rapporteren.

- Helium bevat een groot aantal heuristieken om mogelijke verbeteringen aan te geven bij een fout. Voorbeelden hiervan zijn het herordenen van argumenten en het corrigeren van verkeerd gespelde namen. Verder genereert Helium vele waarschuwingen en hints die helpen om potentiële fouten te voorkomen.

- De compiler kan fouten registreren op een centrale server. Dit is gebruikt tijdens de introductiecursus functioneel programmeren om letterlijk duizenden programma's vast te leggen. Deze programma's zijn geanalyseerd om vast te

stellen welke fouten veel gemaakt worden, en om de heuristieken van de compiler te verbeteren.

• Helium ondersteunt bijna volledig Haskell, met als uitzondering enkele taalconstructies die de kwaliteit van de foutmeldingen negatief beïnvloeden. De meest opvallende taalconstructie die ontbreekt is overloading (type-klassen). In de afgelopen jaren hebben we onderzoek verricht naar het verbeteren van foutmeldingen met overloading. De volgende release van Helium zal type-klassen wel ondersteunen.

• Programma's worden uitgevoerd door de lazy virtual machine (LVM). Bij een exceptie wordt niet alleen de exceptie zelf getoond, maar ook alle waarden die deze exceptie propageren. Dit helpt aanzienlijk bij het debuggen van programma's. Naast standaard excepties zoals ontbrekende patronen, controleert de LVM ook op integer overflow en vormen van oneindige recursie.

• Een simpele, maar effectieve grafische omgeving is beschikbaar voor de Helium compiler (zie Figuur 1). Hierbij kan men automatisch naar de locatie van een fout springen in een editor. Ook worden de verschillende soorten fouten en waarschuwingen in duidelijke kleuren weergegeven.

In de volgende twee paragrafen besteden we eerst aandacht aan de basis van functionele talen, om vervolgens specifiek de Helium compiler te beschrijven.

2 Puur functioneel programmeren

In een pure functionele taal bestaat ieder programma uit een verzameling van definities en expressies. In het volgende voorbeeld wordt een functie gedefinieerd die het kwadraat berekent van zijn argument.

```
kwadraat x = x * x
```

Een interpreter kan nu expressies evalueren die deze definitie gebruiken.

```
> kwadraat (4 + 4)
64
```

Expressies in dergelijke talen gedragen zich als wiskundige formules. De betekenis van een wiskundige expressie is slechts een waarde, en er is geen verborgen extra effect dat afhankelijk is van de wijze waarop men die waarde afleidt. De expressie `kwadraat 4` beschijft precies dezelfde waarde als de expressie `16` of `XVI`, namelijk het getal zestien. Een geldige expressie kan daarom ook in willekeurige volgorde worden gereduceerd.

```
kwadraat (4+4) = kwadraat 8  = 8*8 = 64

kwadraat (4+4) = (4+4)*(4+4) = 8*(4+4)
                             = 8*8 = 64
```

Dit betekent dat we met pure functionele talen equationeel kunnen redeneren over eigenschappen van algoritmen met standaard technieken uit de wiskunde. Imperatieve programmeertalen als C en Java maken dit zo goed als onmogelijk, omdat zij arbitraire effecten toestaan, zoals bijvoorbeeld het rekenen met pointers. Het voordeel van een declaratieve taal als Haskell is dus dat men zich kan concentreren op de essentie, en niet wordt afgeleid door arbitraire details.

Een ander voordeel voor het onderwijs is de mogelijkheid om nieuwe algebraïsche datatypes te introduceren. Met deze definities kunnen de beschikbare datatypes, zoals getallen en letters, verder worden uitgebreid. Een binaire boom met waarden in de bladeren kan als volgt worden opgeschreven.

```
data Boom a = Blad a
            | Tak (Boom a) (Boom a)
```

Het nieuwe `Boom` data type bevat waarden van het type `a` en heeft twee constructoren: een `Boom` bestaat ofwel uit een `Blad` met een waar-

de `a`, of een `Tak` met twee deelbomen. Een voorbeeld van een kleine boom met drie getallen is:

```
nummers = Tak (Blad 1)
          (Tak (Blad 2) (Blad 3))
```

Functies over bomen worden inductief gedefiniëerd over de structuur van de constructoren, zoals bijvoorbeeld een functie die de som van alle getallen in een boom berekent, of een functie die de diepte van een boom bepaalt.

```
som (Blad n)    = n
som (Tak t1 t2) = som t1 + som t2

diepte (Blad n) = 0
diepte (Tak t1 t2)
  = 1 + max (diepte t1) (diepte t2)
```

Met behulp van equationeel redeneren kan men nu vrij gemakkelijk allerlei eigenschappen van bomen bewijzen. Zo zal de diepte van een boom altijd kleiner zijn dan het aantal elementen in de boom. In talen als C of Java, waar de bomen met pointers en references zijn geïmplementeerd, zijn dit soort eigenschappen zeer moeilijk te bewijzen. De bovenstaande definities in Haskell zijn bovendien zeer precies: het equivalente programma in bijvoorbeeld Java zal veel meer arbitraire details bevatten, zoals klasse-definities, type-annotaties en *return* statements.

3 Sterke typering

Haskell bevat een geavanceerd typesysteem dat automatisch vele fouten in programma's opspoort. Het systeem is zo krachtig, dat elk getypeerd programma nooit een ongeldige bewerking zal uitvoeren! Alle Haskell waarden zijn verdeeld in verzamelingen die we types noemen. Basistypes zijn bijvoorbeeld gehele getallen (`Int`) en letters (`Char`). Met deze basistypes kunnen we complexere types samenstellen, zoals bijvoorbeeld functies van getallen naar getallen (`Int -> Int`), of bomen die letters bevatten (`Tree Char`). Hier zijn de types van eerder gedefinieerde namen.

```
kwadraat :: Int -> Int
```

```
nummers  :: Boom Int
som      :: Boom Int -> Int
diepte   :: Boom a   -> Int
```

Vooral het type van `diepte` is interressant: de kleine letter `a` geeft aan dat deze functie *polymorf* is in de elementen van de boom. De diepte van een boom is natuurlijk alleen bepaald door zijn structuur en onafhankelijk van het type van de elementen! Dit betekent ook dat we deze functie slechts eenmaal hoeven te definiëren, en niet voor ieder elementtype apart. Het is verbazingwekkend dat een natuurlijk concept als polymorfie door zo weinig programmeertalen goed wordt ondersteund.

Een ander voorbeeld van een polymorfe operatie is functiecompositie, bekend uit de wiskunde, om twee functies te combineren tot één nieuwe functie. In Haskell gebruiken we hiervoor de operator `(.)`, die als volgt is gedefinieerd.

```
(f . g) x = f (g x)
```

Nu kunnen we bijvoorbeeld `kwadraat . (+3)` opschrijven: deze functie neemt een getal, telt er drie bij op, en neemt daar vervolgens het kwadraat van. We hoeven ons echter niet te beperken tot functies over getallen. Het type van functiecompositie is daarom dan ook zeer polymorf:

```
(.) :: (b -> c) -> (a -> b) -> (a -> c)
```

Functiecompositie neemt een functie van een type `b` naar een type `c`, en een functie van `a` naar `b`, en levert een nieuwe functie op van `a` naar `c`. In talen zonder parametrisch polymorfisme is het type van deze functie niet uit te drukken. In Java bijvoorbeeld moet men onveilige type *casts* gebruiken met het `Object` type om functiecompositie te kunnen beschrijven.

Voor iedere Haskell expressie kan automatisch een type worden afgeleid. Daarom hoeven we nooit het type van een expressie op te geven, maar kan dit worden bepaald door een zoge-

heten type-inferentie algoritme. Elke expressie waaraan geen geldig type kan worden toegekend wordt niet geaccepteerd door de compiler: dit soort expressies hebben nu eenmaal geen zinvolle waarde. Bijvoorbeeld: met de operator `(==)` kan op gelijkheid getest worden, maar men kan geen appels met peren vergelijken.

```
> 1 == 'a'
error: cannot compare Int with Char
```

Wij geloven dat sterke typering essentieel is voor goed informatica onderwijs: het volgen van een stricte type discipline leidt tot heldere en goed gestructureerde programma's. Bovendien vangt sterke typering ook vele logische fouten af: denkfouten leiden bijna altijd tot typeringsfouten!

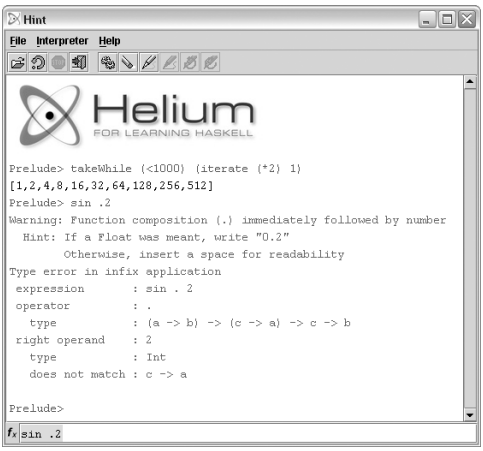
Typesystemen kunnen ook in de weg zitten, zoals elke Java programmeur zal kunnen beamen. Haskell's typesysteem is echter dermate expressief dat dit vrijwel nooit voorkomt. Een belangrijk onderdeel daarvan is de ondersteuning voor polymorfe functies, zoals functiecompositie, en polymorfe datastructuren, zoals `Boom a`. Helaas zijn de foutmeldingen van de bestaande Haskell compilers dikwijls cryptisch van aard en moeilijk te begrijpen. Een voorbeeld van een cryptische melding van de Haskell interpreter Hugs is:

```
> diepte 1
ERROR - Illegal Haskell 98 class
        constraint in inferred type
* Expression : diepte 1
* Type       : Num (Boom a) => Int
```

De Helium compiler is speciaal ontworpen om dit soort cryptische meldingen te voorkomen. In de volgende paragrafen worden verschillende technologische aspecten besproken.

4 De Helium compiler

De twee belangrijkste Haskell implementaties zijn de interpreter Hugs en de Glasgow Haskell Compiler (GHC). De kleinschalige Hugs interpreter is vooral geschikt bij het ontwikkelen van programma's, en wordt veelal gebruikt in het onderwijs. GHC is een "industrial strength" compiler waarmee zeer efficiënte code gegenereerd kan worden, en die tal van uitbreidingen op de taal ondersteunt. In beide gevallen laten de foutmeldingen te wensen over, en hiervoor vormt de Helium compiler een alternatief.



Figuur 1: De Helium interpreter

Het ontwerp van de compiler is afgestemd op het geven van goede foutmeldingen, en verzamelde informatie wordt tijdens het compilatieproces dan ook zorgvuldig bewaard. Helium maakt verder gebruik van een nieuw type-inferentie algoritme dat gebaseerd is op het verzamelen en oplossen van constraints. Deze benadering maakt het mogelijk om niet alleen te constateren dat er een fout gemaakt is, maar ook op zoek te gaan naar de oorzaak hiervan. Maar de compiler onderscheidt zich vooral in de mate waarin er geanticipeerd wordt op veel gemaakte fouten. In de volgende paragrafen kijken we in detail naar de verschillende soorten waarschuwingen en foutmeldingen van Helium.

5 Waarschuwingen en suggesties

De Helium compiler heeft naast echte foutmeldingen ook een groot arsenaal aan waarschuwingen en suggesties. In het algemeen moet men hier zeer voorzichtig mee zijn: niets zeggen kan beter zijn dan het geven van een verkeerde suggestie! Het registratiesysteem kwam hier zeer van pas om vast te stellen wat de meest voorkomende fouten zijn die studenten maken, en de waarschuwingen en suggesties van Helium zijn veelal gebaseerd op deze gegevens. In paragraaf 8 presenteren we het registratiesysteem in meer detail.

Het volgende voorbeeld illustreert het nut van de suggesties en waarschuwingen die door Helium worden gerapporteerd:

```
myFilter::(a -> Bool) -> [a] -> [a]
myFilter p [] = []
myFIlter p (x:xs) =
        if p x
         then x : myFilter p xs
         else myFilter p xs
```

Het bovenstaande programma wordt door iedere andere Haskell compiler geaccepteerd zonder enige waarschuwing. Helium daarentegen ziet een aantal potentiële problemen:

```
(4,1): Warning: Tab character encountered;
        this may cause problems with layout
  Hint: Configure your editor to
        replace tabs by spaces
(3,1) : Warning: Missing type signature:
         myFIlter :: (a -> Bool) -> [a] -> [a]
(2,10): Warning: Variable "p" is not used
(2,1), (3,1): Warning: Suspicious adjacent
  functions "myFilter" and "myFIlter"
```

We zien dat Helium meerdere waarschuwingen kan geven met precieze locaties in de broncode: regel en kolom. De laatste waarschuwing laat ook zien dat Helium meerdere locaties kan rapporteren die bijdragen aan een melding. In een ontwikkelomgeving kan dit gebruikt worden om tussen de verschillende locaties te navigeren.

De eerste waarschuwing is een typische melding voor een compiler in het onderwijs: fouten in de layout van Haskell programma's vanwege onzichtbare tabs zijn een bron van frustratie voor studenten. De tweede melding moedigt studenten aan om type signaturen op te schrijven bij de functies die ze definiëren. Hiermee stimuleren we een goede gewoonte, en wordt de student gedwongen na te denken over zijn opgeschreven functies. Zo'n waarschuwing hoort juist thuis in een compiler gericht op het onderwijs. In dit specifieke voorbeeld is de waarschuwing overigens veroorzaakt door een schrijffout. De laatste waarschuwing wijst deze moeilijk te vinden fout aan: de naam van de functie in de laatste vergelijking is verkeerd gespeld. Uit ons registratiesysteem blijkt dat dit soort fouten geregeld voorkomen, maar pas na relatief lang *debuggen* gevonden worden.

Helium berekent de *minimale bewerkingsafstand* tussen twee definities die elkaar opvolgen. Een bewerkingsafstand is gespecificeerd als het aantal basisbewerkingen dat men moet uitvoeren om de ene naam in de andere te veranderen. In het bovenstaande geval is de afstand slechts één bewerking: de letter `i` moet in een hoofdletter veranderd worden. Wanneer de afstand klein genoeg is neemt Helium aan dat er een schrijffout is gemaakt en wordt er een waarschuwing gegeven.

Hetzelfde mechanisme wordt gebruikt om schrijffouten in de definitienamen te ontdekken. Hier is een ander programma uit het registratiesysteem:

```
maxLen :: [String] -> Int
maxLen = maximun (map length xs)
```

Dit programma bevat twee ongedefinieerde variabelen:

```
(2,10): Undefined variable "maximun"
  Hint: Did you mean "maximum" ?
(2,30): Undefined variable "xs"
```

Helium meldt ons dat `maximun` wel erg lijkt op een andere naam die wel gedefinieerd is, name-

lijk `maximum`.

Een andere veel gemaakte fout van studenten is om floating-point getallen verkeerd op te schrijven.

```
test = sin .2
```

Qua syntax is dit een goed Haskell programma omdat de punt (`.`) gelezen kan worden als functiecompositie. De interpreter Hugs geeft dan ook een verwarrende foutmelding:

```
ERROR "lex1.hs" (line 1):
  Unresolved top-level overloading
* Binding             : test
* Outstanding context : (Floating b,
                         Num (c -> b))
```

Deze foutmelding is het gevolg van de impliciete overloading van integer constanten. Helium geeft natuurlijk ook een typefoutmelding, maar pas na de waarschuwing dat het ongebruikelijk is om een punt voor een getal neer te zetten.

```
(1,13): Warning:
  Function composition (.)
  immediately followed by number
  Hint: If a Float was meant, write "0.2"
        Otherwise, insert a space for
        readability

(1,13): Type error in infix application
 expression     : sin . 2
  operator      : .
  type          : (a -> b) -> (c -> a) -> c -> b
 right operand  : 2
  type          : Int
  does not match: c -> a
```

Natuurlijk zijn zulke suggesties veel waard voor de beginnende Haskell programmeur. Studenten zullen zich sneller de syntax eigen maken, en kunnen dus meer aandacht besteden aan de werkelijke leerdoelen

6 Syntaxfouten

Zoals het voorgaande voorbeeld laat zien, kan men snel (te) veel tijd besteden aan het aanleren van syntax, in plaats van de essentiële concepten van puur functioneel programmeren. Helium probeert daarom uitgebreide syntactische foutmeldingen te geven; dit heeft slechts weinig prioriteit in productie compilers. Een programma wordt eerst lexicaal geanalyseerd door de balancering van haakjes te controleren. Hierbij worden vele fouten afgevangen die anders leiden tot cryptische syntaxfouten.

```
test :: [(Int, String)]
test = [(1, "one"), (2, "two"), (3, "three")
```

In het bovenstaande voorbeeld is de programmeur vergeten de lijst af te sluiten met een sluitteken (`]`). Omdat de balancering van haakjes nauwgezet wordt bijgehouden kan Helium hiervoor een exacte foutmelding produceren:

```
(2,8): Bracket '[' is never closed
```

Hier is een duidelijk verschil zichtbaar met de melding van de GHC compiler. Deze houdt geen kolom informatie bij en is zeer imprecies in de locatie van de fout. Verder wordt er ten onrechte gesuggereerd dat dit te maken heeft met verkeerde indentatie:

```
syn3.hs:3: parse error
  (possibly incorrect indentation)
```

Voor het bovenstaande voorbeeld geeft de interpreter Hugs een interessante melding omdat het naar een niet bestaand sluitteken verwijst. Dit teken is door de compiler *zelf* toegevoegd om de layout van Haskell te analyseren.

```
ERROR "syn3.hs" (line 3):
  Syntax error in expression
  (unexpected '}', possibly due to bad layout)
```

De grammaticale structuur wordt geanalyseerd met behulp van geavanceerde parsertechnologie, genaamd Parsec. Deze technologie houdt niet alleen de positie van syntaxfouten bij, maar ook alle syntax-constructies die op dat punt van de invoer geldig waren geweest.

```
remove :: Int -> [Int] -> [Int]
remove n [] = []
remove n (x:xs)
   | n = x     = rest
   | otherwise = x : rest
 where rest = remove n xs
```

In de vierde regel van het bovenstaande voorbeeld worden `n` en `x` vergeleken met `=`, terwijl hiervoor de operator `==` gebruikt had moeten worden. Helium merkt de fout op zodra de tweede `=` wordt gezien.

```
(4,16): Syntax error:
    unexpected '='
    expecting expression, operator,
    constructor operator, '::', '|',
    keyword 'where', or end of block
    (based on layout)
```

Contrasteer dit met de melding van de GHC compiler:

```
syn4.hs:4: parse error on input `='
```

Helium laat duidelijk de locatie van de fout zien, terwijl het in de foutmelding gegeven door GHC onduidelijk is welke van de twee `=` tekens verantwoordelijk is voor de fout. Verder laat Helium zien welke sytax constructies wel geldig zijn op dat punt van de invoer.

7 Typeringsfouten

Het registratiesysteem laat zien dat veruit de meeste fouten die gemaakt worden typeringsfouten zijn. Zoals gezegd infereert Helium automatisch de types van geldige expressies, en wordt er een melding gegeven indien er een fout gemaakt is. Echter, om extra ondersteuning voor foutmeldingen te kunnen bieden wijkt het type-inferentie mechanisme sterk af van de traditionele benadering. De analyse is geformuleerd als een constraint probleem om zoveel mogelijk informatie vast te houden tijdens het inferentieproces. Aan de hand van een verzameling constraints gegenereerd voor een programma wordt er een graaf opgebouwd die de samenhang tussen expressies en types weergeeft. Inconsistenties in de graaf worden opgelost door het inzetten van een verzameling heuristieken om veel voorkomende patronen te detecteren, en hier geschikte meldingen voor te geven.

Eén heuristiek voor het oplossen van een type-inconsistentie is het tellen van het aantal constraints die een bepaald type prefereren. Als er bijvoorbeeld drie indicaties zijn dat een expressie type `Int` moet hebben, maar slechts één indicatie dat het een `Bool` moet zijn, dan zal de foutmelding zich concentreren op de uitzonderlijke `Bool`.

```
makeEven :: Int -> Int
makeEven x = if even x then True else x+1
```

In het bovenstaande programma bevatten de `if` takken een `Bool` en een `Int` expressie. Vanwege de typesignatuur heeft Helium meer aanwijzingen dat dit `Int` moet zijn, en de foutmelding geeft weer dat de `True` expressie fout is:

```
(2,29): Type error in then branch
expression     : if even x then True else x + 1
term           : True
 type          : Bool
 does not match: Int
```

Traditionele type-inferentie neemt dergelijke constraints niet mee en zal bijvoorbeeld een foutmelding geven over de andere tak. Het globaal oplossen van de verzameling constraints voorkomt een van-links-naar-rechts afwijking die typisch voorkomt in standaard unificatie-algoritmen.

Met behulp van de typeringsgraaf kan Helium een aantal strategieën toepassen om inconsistenties op te lossen. Als dit lukt kan Helium een suggestie geven om het inzicht in de gemaakte fout te vergroten. Om misleidende suggesties te voorkomen geven we alleen een suggestie als deze een unieke oplossing biedt. Voorbeelden van strategieën zijn het verwisselen van functie argumenten die in de verkeerde volgorde gegeven zijn, of het invoegen van een vergeten argument. Neem bijvoorbeeld het volgende programma:

```
test = map [1..10] even
```

De hogere-orde functie `map` verwacht twee argumenten: een functie en een lijst. De student heeft de argumenten van `map` echter in de ver-

keerde volgorde gezet. Helium anticipeert op deze veelvuldig gemaakte fout.

```
(1,8): Type error in application
 expression     : map [1 .. 10] even
 term           : map
  type          : (a -> b)-> [a]          -> [b]
  does not match: [Int]   -> (Int -> Bool)-> c
 probable fix   : re-order arguments
```

Helium gebruikt een variant van de minimale bewerkingsafstand om te bepalen hoe expressies veranderd kunnen worden om de type-inconsistenties in de typeringsgraaf op te lossen. Als dit een unieke oplossing biedt bij weinig bewerkingen, zoals het herordenen van functie argumenten, dan geeft Helium een suggestie. Merk ook op dat het typesignatuur van `map` is gegeven in de foutmelding, en dat dit mooi is uitgelijnd met het geïnfereerde type. GHC verwijst slechts naar een willekeurig argument van `map`:

```
tp4.hs:1:
 Couldn't match 'a -> b' against '[t]'
    Expected type: a -> b
    Inferred type: [t]
 In an arithmetic sequence: [1 .. 10]
 In the first argument of 'map',
    namely '[1 .. 10]'
```

Een andere belangrijke strategie beschouwd *siblings*: dit zijn semantisch gerelateerde functies met enigszins afwijkende types. Zo zijn er bijvoorbeeld twee functies beschikbaar om het maximum te bepalen:

```
max     :: Int -> Int -> Int
maximum :: [Int] -> Int
```

De ene werkt op twee getallen, de andere op een lijst met getallen. Omdat deze functies zo verwant zijn aan elkaar maken we er siblings van. Op een soortgelijke manier relateren we ook integer constanten aan floating-point getallen. Helium ondersteunt speciale directieven waarmee dit soort sibling paren kunnen worden opgeschreven:

```
sibling   max , maximum
```

Het volgende voorbeeld laat de sibling strategie in actie zien. Het programma is geschreven door een student als onderdeel van het werkcollege. Meerdere studenten maakten daar dezelfde fout.

```
maxLength :: [String] -> Int
maxLength xs = max (map length xs)
```

Deze definitie is bijna correct, alleen heeft de student de verkeerde functie gebruikt om het maximum te bepalen. De Hugs interpreter geeft de volgende melding:

```
ERROR "A.hs":2 -
    Type error in explicitly typed binding
*** Term           : maxLength
*** Type           : [String] -> [Int] -> [Int]
*** Does not match : [String] -> Int
```

Omdat het type van `max` normaal gesproken overloaded is kan Hugs deze definitie typeren, al komt het geïnfereerde type niet overeen met de opgeschreven typesignatuur. De GHC compiler is al iets beter in zijn foutmelding:

```
A.hs:2:
    Couldn't match 'Int' against 'a -> a'
        Expected type: Int
        Inferred type: a -> a
    Probable cause:
        'max' is applied to too few arguments
        in the call (max (map length xs))
    In the definition of 'maxLength':
        max (map length xs)
```

Omdat Helium een typeringsgraaf gebruikt, kan deze opmerken dat de sibling functie van `max` de inconsistentie uniek kan oplossen. Helium geeft daarom de suggestie om `maximum` te gebruiken:

```
(2,16): Type error in variable
 expression       : max
   type           : Int   -> Int -> Int
   expected type  : [Int] -> Int
 probable fix     : use maximum instead
```

Deze melding is niet alleen makkelijk te begrijpen vanwege de sibling, maar ook omdat het type van `max` vermeld wordt.

8 Het registratiesysteem

Tijdens de introductiecursus functioneel programmeren aan de Universiteit Utrecht hebben we een registratiesysteem ingezet om studenten-

programma's op te slaan. Het doel hiervan is om inzicht te krijgen in het typische programmeergedrag van beginnende programmeurs, en om met het verkregen inzicht de compiler nog beter aan te laten sluiten bij de behoeften van deze doelgroep. Gedurende de looptijd van het vak is ieder programma dat gecompileerd wordt vanaf de studentenmachines opgeslagen in een database. Expressies die tussentijds in de Hint interpreter worden geëvalueerd zijn buiten beschouwing gelaten. Dit heeft in de afgelopen twee jaar geleid tot een database van ruim 60.000 programma's, zowel correcte als incorrecte. Om dit te bereiken hebben we de Helium compiler voorzien van de mogelijkheid om contact te zoeken met een centrale server. Deze server legt de opgestuurde programma's vast.

Om de samenhang tussen de programma's in deze database te behouden wordt bij ieder programma historische informatie opgeslagen. Dit bestaat uit de naam van de student, het tijdstip van compilatie, en het versienummer van de compiler. Dit maakt het mogelijk om de voortgang van een individu te traceren en analyseren. Vanwege de precisie in de registratie hebben we enkele maatregelen genomen om de privacy van de studenten te kunnen garanderen. Ten eerste zijn alle studenten vooraf ingelicht over het experiment, en bestond er de mogelijk om niet geregistreerd te worden. Ook is er afgesproken dat de geregistreerde gegevens in geen enkel opzicht van invloed zouden zijn bij de beoordeling van het praktikum. Tenslotte is de database na afloop van het vak geanonimiseerd.

De opgeslagen gegevens vragen om een uitgebreide analyse, en hier wordt momenteel aan gewerkt. Om toch een idee te geven over de mogelijkheden van zo'n analyse presenteren we enkele gegevens uit de dataset van het cursusjaar 2003/2004. De volgende tabel geeft per week het aantal registraties weer.

wk.	6	7	8	9	10	11	12	13
reg.	3511	3649	3263	901	6279	1590	1962	2157

De negende week was lesvrij, en de praktikum deadlines lagen aan het einde van week 10 en week 13 (zoals wellicht ook is af te leiden uit de data). Alle registraties kunnen we indelen in compilatie categorieën: deze komen overeen met de verschillende fasen in een compiler die een programma moet doorlopen. We geven percentages per categorie voor week 6, en voor het gehele praktikum.

compilatie categorie	*week 6*	*totaal*
lexicale fout	3%	3%
syntaxfout	14%	9%
statische fout	8%	10%
typeringsfout	30%	32%
compileerbaar	45%	46%

Gelukkig blijkt bijna de helft van de programma's "correct" te zijn. Hieronder vallen echter ook programma's die een denkfout bevatten en dus semantisch gezien onjuist zijn, en programma's waarin Helium potentiële fouten heeft ontdekt. De significante verschuiving van het percentage syntactisch incorrecte programma's bevestigt het vermoeden dat de eerste hindernis om een programmeertaal te leren het bekend raken met de syntax is. Tenslotte geven we een overzicht van de verdeling van het soort statische fouten. Een programma kan meerdere van dit soort fouten bevatten.

statische fout	*aantal*	*(%)*
ongedefinieerde variabele	2240	55,46%
ongedefinieerde constructor	377	9,33%
typesignatuur zonder functie	368	9,11%
ariteit bij constructor	223	5,52%
ariteit bij functiedefinitie	209	5,17%
ongedefinieerde typeconstr.	196	4,85%
dubbele definitie	157	3,89%
overige fouten	269	6,66%

Opvallend genoeg is het refereren aan een onbekende variabele met afstand de meest gemaak-

te vergissing. Dit onderstreept nogmaals de importantie van het berekenen van een minimale bewerkingsafstand voor ongedefinieerde variabelen.

9 Conclusie

We hebben laten zien dat functionele talen (zoals bijvoorbeeld Haskell) zeer geschikt zijn om de essentie van programmeren te onderwijzen. Dat declaratieve talen en onderwijs goed samengaan wordt tevens aangetoond door het Pan# project. In deze leeromgeving kunnen functionele afbeeldingen en animaties beschreven worden door middel van declaratieve definities en functionele abstracties. Dit systeem wordt gebruikt op middelbare scholen in Amerika om de basis van algebra uit te leggen op een simpele en interactieve manier. Een tweede voorbeeld is de DrScheme programmeeromgeving, waarmee studenten stapsgewijs kennis kunnen maken met de functionele taal Scheme.

Helium kan een waardevolle ondersteuning bieden bij het leren van de programmeertaal Haskell. De gerapporteerde foutmeldingen zijn duidelijk en precies, en door het geven van waarschuwingen wordt een goede programmeerstijl bevorderd. Bovendien wordt er geanticipeerd op een aantal karakteristieke fouten. De compiler is vrij beschikbaar op de Helium website:

http://www.cs.uu.nl/helium

Helium richt zich uitsluitend op het leren van de programmeertaal Haskell. Voor meer geavanceerde toepassingen zal men de overstap naar een compiler moeten maken met meer functionaliteit en bibliotheken, zoals bijvoorbeeld GHC. Hier heeft men ook de beschikking over de wxHaskell library om grafische applicaties te maken. Deze groots opgezette bibliotheek voor grafische user interfaces is gebaseerd op wxWidgets, een uiterst succesvol cross-platform en open-source GUI framework. Na het leren van de essentiële principes van programmeren met Helium, kan men zo aan de slag om grote interactieve applicaties te ontwikkelen.

Onze dank gaat uit naar Arjan van IJzendoorn, die tweeënhalf jaar geleden is begonnen met het ontwerpen van de compiler. Jurriaan Hage heeft een belangrijke rol gespeeld in het opzetten van het registratiesysteem, en het uitvoeren van de beschreven experimenten. Tenslotte bedanken we Rijk-Jan van Haaften en Arie Middelkoop voor hun geleverde bijdrage aan het project.

Referenties

Findler, Robert Bruce *et al.* (2002). DrScheme: A programming environment for Scheme. *Journal of functional programming*, **12**(2), 159–182.

Heeren, Bastiaan, & Leijen, Daan. (2004). Gebruikersvriendelijke compiler voor het onderwijs. *Informatie*, **48**(6).

Heeren, Bastiaan, Leijen, Daan, & van IJzendoorn, Arjan. (2003a). Helium, for learning Haskell. *Pages 62 – 71 of: ACM sigplan 2003 Haskell workshop*. New York: ACM Press.

Heeren, Bastiaan, Hage, Juriaan, & Swierstra, S. Doaitse. 2003b (Aug.). Scripting the type inference process. *International conference on functional programming (ICFP'03)*.

Leijen, Daan. (2004). wxHaskell. *ACM sigplan 2004 Haskell workshop*. Snowbird, Utah, USA: ACM Press.

Peterson, John. 2002 (October). A language for mathematical visualization. *Proceedings of FPDE'02: Functional and declarative languages in education*.

Peyton Jones, Simon, & Hughes, John (eds.). 1998 (Feb.). *Report on the language Haskell'98*.

Athena, a large scale programming lab support tool

Anton Jansen, University of Groningen, The Netherlands

Abstract

Providing a programming lab to a large group of students requires a lot of effort. Often in these cases additional staffing is required to provide students with enough feedback on their work. However, due to resource constraints this is not always possible. This paper presents Athena, a lab support tool that reduces the effort required for programming labs and provides students with adequate and timely feedback. Athena reduces the need for additional staffing and provides students with a better learning experience. Furthermore, it allows for new educational forms for large student groups like programming lab exams.

Keywords: Programming labs, automated testing

1 Introduction

A programming cours is often supported by one or more programming labs. The exercise(s) of the labs are meant to teach students the skill of programming and the concepts of a certain programming paradigm. For large groups of students, the lab part of a programming course becomes laborious. This is mainly due to the fast amount of review work that needs to be done to provide students with feedback on their work.

In this paper, Athena is presented. Athena is a system that supports large scale programming labs and reduces the effort required to run such labs. Athena achieves this by automating certain laborious administrative tasks and by providing automated feedback towards the students on their work. The benefits of Athena are twofold: it reduces the need for additional staffing and provides a better learning experience for students.

The contribution of this paper consists of three parts. First, it introduces the Athena system and argues for the two aforementioned benefits of improved learning and reducing effort. Second, it provides considerations and guidelines how Athena can be integrated into programming courses. Third, it presents an in-depth evaluation of the pro's and con's of a support system like Athena.

The rest of this paper is organized as follows. First, a closer look is taken at programming courses, programming labs, and their laborious tasks. Next in section 3, Athena is introduced. Section 4 explains how Athena can be used in programming courses. After which the paper presents an overview of the lessons learned in section 5 and concludes in section 6 with conclusions.

2 Programming courses

Programming can be learned in many different ways. In this section, a look is taken at supervised learning in the form of programming courses. Two different strategies of teaching programming courses can be identified: the algorithm perspective and the component perspective.

In the algorithm perspective, the focus is on learning students to think in steps to build up an algorithm for a particular problem. On the other hand, in the component perspective the focus is on how predefined functionality (e.g. algorithms) in the form of components can be composed together to form an application.

However, both strategies try to achieve similar goals. Programming courses are not meant to learn a particular programming language to a student, but more the concepts behind a particular programming language paradigm. For example, in Object Oriented Programming (OOP) the concepts of methods and inheritance, in functional programming the concept of folding.

Programming courses additionally teach students to make transformations from the problem domain (the real world) to the solution domain (a programming paradigm). For example, analytic, problem solving, and divide & conquer techniques can be learned to ease this difference in worlds between

computers and the real world.

Furthermore, programming courses try to develop the student's sense for good and bad programming practices. This includes considerations of aesthetics, quality, and trade-offs made. Students need to become aware of these issues and have the ability to discuss and reason about them.

Learning how to program is a difficult and time-consuming process. The main reasons for this are: (1) the abstract nature of, (2) the required precision for, (3) and the skill required, to use concepts of programming languages. So how can we deal with these problems in a programming course?

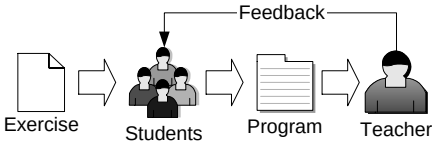


Figure 1: Programming labs

Many teachers use the solution to have a series of programming labs associated with their programming course. In these programming labs, programming exercises are used to train students in programming. The exercises are meant to train the students in putting the theoretical knowledge of programming concepts and combinations of them into practice. Often there is a need to have many small exercises to train concepts in isolation, before complex and particular combinations are tried.

In figure 1, the process of a programming lab is depicted. First, one or more exercises are defined by a teacher. They serve as the main input for the students. It is the student task to make a program as described in the exercise. This program is communicated to a teacher, which reviews and/or grades the program. The remarks of the teacher are communicated back to the students as feedback.

Reviewing and/or grading student programs for these exercises is a time-consuming job. The main reason for this is the sheer amount of code that needs to be examined. For example, 60 students are more than capable of writing over 7500 Lines Of Code (LOC) in a time span less than 3 hours for some small exercises. For large groups of students, the amount of review work can become so great that it prohibits a timely feedback from the teacher(s) towards the students. Consequently, the learning experience of the students becomes less than optimal in these cases.

For large groups of students, there is a clear need for more staffing to prevent such situations. However, due to resource constrains this is not always possible or economical feasible. Consequently, a different approach is needed for large groups, which reduces effort for a teacher, but does not compromise the learning experience of the student.

3 Athena

Athena is a computer system to support large scale programming labs. It reduces the required effort for a teacher to run these labs, while enriching the students learning experience. Figure 2 presents an overview of how programming labs with the support of Athena work. The main difference with the process of normal programming labs (see figure 1) is the additional feedback to students and filtering of information to the teacher.

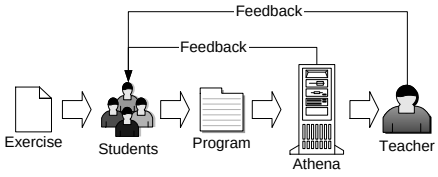


Figure 2: Programming labs with Athena

In the rest of this section, different aspects of Athena are presented. First, section 3.1 outlines the basic working process of Athena and it's supporting tools. Next, section 3.2 takes a closer look at the automated testing facilities of Athena, which creates the additional feedback towards the students.

3.1 Athena system

The basic working process of Athena (see figure 2) consists of students submitting their work to the Athena system using the *submission client* (see figure 3). The submitted work is automatically tested by the *arbiter*, which provides the additional feedback to the students using the *student web interface*. The teacher can examine the submitted work and configure the automated testing by the use of the Athena *management tool* (see figure 4). Following is a more detailed description of these supporting tools and their function:

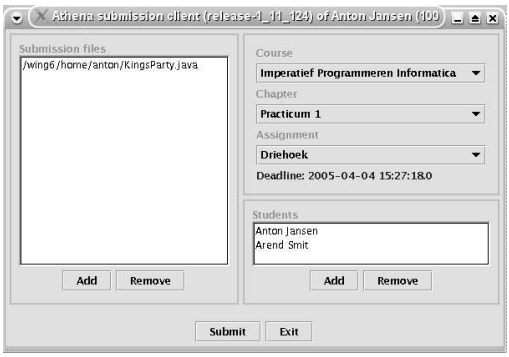


Figure 3: The submission client

Submission client The submission client (see figure 3) is a stand-alone Java application, which students use to submit their work to the Athena system. The student can select the files or directories containing their solution. To reduce garbage a filter can be defined, which filters out unwanted material (e.g. binaries). Optionally, in the case of group work, the student can select on or more fellow students that are responsible for the submitted work as well. This relieves the staff of the burden to maintain a proper group administration. For large groups of students this turns out to be a time-consuming and error-prone task, as groups tend to change quite often due to sickness, dropouts, and arguments among students.

Arbiter The arbiter is the heart of the Athena system, as it judges the submissions automatically. For each exercise that should be automatically tested, a testset consisting of one or more tests is defined in Athena. The arbiter executes these various tests and collects the result in a judgement about the submission. An elaborate description of the various means to test is presented in section 3.2.

Student webinterface Students receive the feedback from the Athena system through the *student webinterface* [Athena website]. This web-based interface provides a student with an overview of his or her submissions and their status in the Athena system. Futhermore, the judgement (created by the arbiter) can be examined. It includes detailed information about the executed tests, their results, expected results, and additional hints defined by the teacher.

Management tool The management and configuration of the various elements of Athena is done in the management tool (see figure 4). This tool allows teachers to configure the Athena system to their needs and manage their programming courses. For example, managing deadlines, creating and modifying courses and exercises, and printing and reporting are done using the management tool.

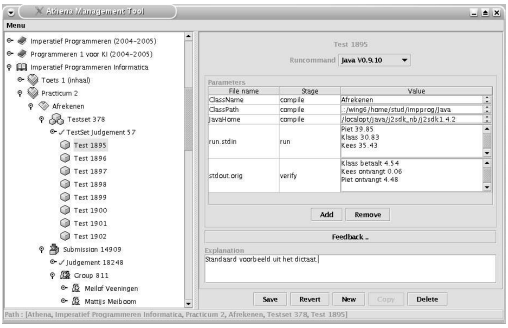


Figure 4: The management tool

3.2 Automated testing

The automated testing system of the arbiter is the heart of the Athena system. It generates the additional feedback for the students (see figure 2), thereby reducing the burden on the staff. Athena provides a testing *environment* as opposed to a single testing *framework*, as Athena needs to be pro-

gramming language and platform independent. In this testing environment, different testing frameworks are specified for the different combinations of programming languages and platforms.

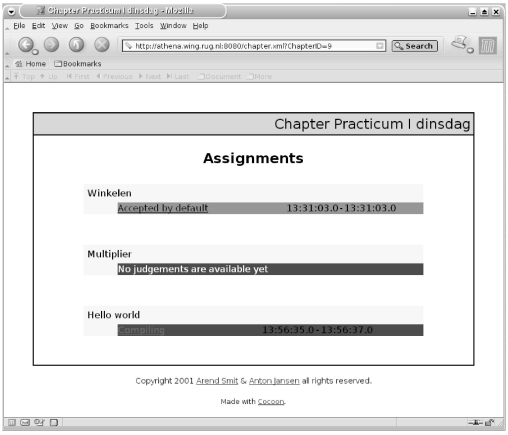


Figure 5: Student webinterface

Teachers configure a testing framework in the management tool to define an automated test for their particular exercise. Testing frameworks have been defined and used for multiple programming languages, which include frameworks for Modula-3, Pascal, Java, and C.

The test environment of Athena provides an infrastructure for the testing frameworks to report their findings to the system and optionally back to the student. Consequently, the Athena is completely programming language independent and therefore could be used for functional and logic programming languages as well.

Various testing techniques can be used in Athena. The following list gives an overview of the techniques, which have been used in Athena:

Don't do it From experience, we have learned that it is sometimes convenient not to test at all. Consequently, the Athena system then becomes an administration tool, which is specialized for programming courses.

Compile The work of the student is tested with a compiler to see whether it compiles or not. Optionally, warnings of the compiler can be allowed or disallowed. Surprisingly enough, many submissions fail to pass this test, due to small mistakes that are very hard to detect by a human.

Run The compiled work of the student is run to see whether it does not generate any run-time anomalies (e.g. segmentation faults, out-of-bounce exceptions etc.). Furthermore, it can be tested whether the program ends in a predefined time, to have an indication for end-less loop constructs.

Output The generated output of a student program on a given input can be automatically tested as well. The used options in Athena frameworks for output testing include:

- **Textual** The output of the student program is textual compared to the output of a reference program or a predefined output. Texts can be compared in numerous ways, the one used often in Athena is a context difference, which ignores whitespace (i.e. spaces and tabs).
- **Numerical** When the output consists of numbers, numerical tests can be made. The numbers of the student can be compared against predefined numbers or numbers generated by a reference solution. Depending on the absolute or relative difference the test can bet set to fail or succeed.
- **Dedicated** A special made program could be used to test the student program and interpreted the generated output in a domain specific way. For example, an application that automatically tests whether a circle is drawn on the screen by the application.

Performance The performance of the student program can be measured in time, memory usage, etc. In the use of Athena, this is primarily used to automatically test whether the student uses a particular efficient algorithm (e.g. quicksort).

Student The idea is here to let the students make the test themselves in a predefined testing framework. For example, Athena has a JUnit [JUnit] testing framework. This allows students to test their own Java application in a uniform way. The educational benefit is that students are forced to think

about testing, which is an inherent part of software development. The teacher now only needs to check the coverage of the testset of a student to ensure functional correctness.

4 Courses with Athena

In the previous section, the Athena system was introduced. This section presents how the Athena system can be integrated into programming courses and the consequences of this integration.

Although testing is regarded in software engineering as a crucial process to insure quality, most programming courses regard it as a separate topic, which lies often out of scope. If the automated testing facilities of Athena are used in a programming course, testing becomes an inherent part of the course. However, for automated testing to work in a course, a couple of issues need to be dealt with:

Interactions with the environment This is the most constraining, hard, and often an impossible aspect to achieve. For automated testing the interaction between the student program and its environment needs to be specified. Otherwise, the testing cannot be automated.

Transforming an idea to an exercise text that defines these required interactions proves far from trivial. The level of detail required prohibits exercises using many different types of interactions. Furthermore, the "catch" of some exercises lies in these interactions. Specifying them in the exercise text destroys the purpose of the exercise, as it gives the "catch" away.

Several strategies exist to ease interaction with the environment. The easiest and most often employed one is to provide examples of the intended interaction. A more specialized strategy is to provide the student with a framework, which already takes care of these issues. However, learning the framework takes time and requires knowledge sometimes not available to the students (e.g. in a starters programming course), therefore this strategy is not always a viable option. Another strategy is not to specify these interactions beforehand. This prevents testing of predefined tests, but still allows students to make their own testset (see 3.2).

Accuracy The texts of an exercise need a high degree of accuracy to prevent ambiguity. As ambiguous situations makes automated testing for these cases near to impossible. Furthermore, to deal with the issue of the previous point the interaction with the environment needs to described accurate and in detail.

Covering and incremental testset To benefit from the automated testing facilities of Athena a good testset is required. A good testset for exercise covers the various issues of an exercise. Furthermore, it is defined in an incremental way, such that first some trivial cases are tested, after which the more complex ones come. This is needed to make the relationship between the tests and the student program clear, thereby providing a smooth learning path. Often tests are made to check whether the wrong conceptual solution has been chosen. Feedback associated with these tests are defined to include instructions or remarks what the conceptual mistake likely would have been.

Reference solution Determining the coverage and suitability of a testset is difficult. Furthermore, experience learns that many mistakes are made in the definition of a test. A reference solution is therefore a crucial tool to test the testset and to provide validation of the suitability of the exercise.

The issues presented make the initial effort to define an exercise for automated testing significant higher than without. An automated testing environment like Athena pays-off in the correction phase. In courses using Athena, only submissions that have *passed* the testset are corrected and graded by the teacher. This significantly reduces the effort, as the teacher can assume the functional correctness of the student work.

Athena allows for new forms of benchmarking student performance. Programming exams can be held in which students get a fixed amount of time (e.g. 3 hours) to solve as many exercises as they can. The focus of the exam is to test the student's skill in *using* and applying a programming language, rather

than the theoretical knowledge about a language tested with traditional paper and pencil exams.

5 Lessons learned

Athena has been in use since 2001 for 3 or 4 different courses each year. Over 1000 different students from 4 different faculties have used the system so far. In this section, the experiences and lessons learned during this time are presented. First, the positive and negative sides are presented from the perspective of a student, after which the same is done for the teacher. From a student perspective, there are the following observations:

- **Hard, competitive** Submitted work of a student is only graded once it *passes* the automated tests. This makes the system very harsh towards students, as there is no consideration of (inadequate) work that has not passed the testset. This hardness also creates a competitive atmosphere, as students compare their results, which are instantly available.

- **Fraud** The hardness of the automated testing makes the temptation to fraud bigger. The large number of students for which Athena is used complicates the detection of such cases.

+ **Precise working attitude** The automated testing part of Athena encourages students to pick up a precise working attitude. The feedback of the testing systems confronts the student with mistakes of their program, independent of how small and trivial they might be.

+ **Objective** Athena's automated testing is objective, as opposed to the manual checking of functional correctness by a group of teachers. Each teacher has it's own way of finding flaws within a program. Consequently, this can leads to situations where a certain type of fault is detected by one, but not by the others. Hence, the determination of functional correctness becomes subjective. To fight this subjectivity, considerable effort needs to be spent in harmonizing and communicating functional correctness requirements, something that is not needed in Athena.

+ **Learning experience** The fast feedback of the system provides a better learning experience for students. No longer, they have to wait one or two weeks to get (detailed) feedback on their work. The provided feedback can be *directly* used in the same lab sessions. This prevents situations where a student makes the same systematic mistake during several lab sessions, as the grading process often fails to match the lab sessions pace. Furthermore, the quality of the student work has been *significant* higher than in cases where Athena is not used.

There are also some lessons learned from a teacher perspective:

- **Reduced exercise freedom** As already pointed out in section 4, the use of automated testing constrains potential exercises.

- **Dependability** The automated testing of Athena makes a course very dependent on a specific system environment. Ideally, the development environment of the student matches the testing environment, as a detected fault then can be replicated by the student. Our experience has learned there can arise many subtle differences due to a different mix of compiler and libraries versions, operating systems, access rights, and other system variables. Furthermore, these environments evolve, thereby sometimes requiring evolution of the testset of an exercise as well.

+ **Quality of grading** As the Athena system handles the functional aspect of the student work, the focus of grading shifts from functionality correctness to quality aspects of the work.

+ **Saves time** Although the initial investment is high to setup a course with Athena, the investment pays off for large groups. The automat-

ing of part of the administration and the automated testing saves time. The break-even point lies around groups of 40 students. If a course in can be reused for several years, this number can get as low as 25.

+ **Scalability** An advantage of Athena is its scalability. Courses with over 160 students participating use Athena. The staffing for the weekly labs of these courses consists of three teaching assistants working 8 hours a week. Only additional computers are needed to ensure fast feedback from the automated testing.

6 Conclusions

Programming labs are an essential part of programming courses. The exercises used in programming labs provide students with insight into the workings of the concepts of a programming paradigm. Furthermore, students learn to use a particular programming language, thereby training their programming skills.

An important part of the learning experience of students is the feedback they receive from the teacher on their work on the exercises. For large groups of students providing this feedback can become a problem. The review work becomes so great for a teacher that it no longer can be provided on time. Consequently, the learning experience for students becomes less than optimal.

This paper presented Athena, a support tool for large scale programming labs. Athena automates common administrative tasks like group lists, deadline management, result administration, and printing of student programs. Furthermore, Athena automatically determines the functional correctness of the work of students using automated testing.

The automated testing facilities of Athena improve the student learning experience, as more and timely feedback can be provided to the student. Furthermore, the automated testing significantly reduces the review time of a teacher. Athena takes care of the functional part of the student work. Consequently, a teacher can spend his or her precious time on reviewing the non-functional quality aspects (e.g. coding style and documentation).

Both automation of administrative tasks and the automated testing facilities significantly reduce the required effort from teacher(s) for large groups of students. As the initial effort required to define the necessary tests for automated testing is relative high, Athena starts paying off for groups larger than 40 students. This is especially the case if the course is used for more than a single occasion.

Further work on Athena includes the integration of a fraud detection system (e.g. Moss [Moss]). Especially for large groups of students this will fight the temptation for students to commit fraud. In addition to fraud detection, further work is needed at easing deployment of Athena. Currently, there is no nice installation procedure for the different user applications. This is something that needs to be developed to make Athena easier deployable for others.

References

[Athena website] http://athena.wing.rug.nl:8080

[JUnit] http://www.junit.org

[Moss] http://www.cs.berkeley.edu/˜aiken/moss.html

Denkniveaus bij algoritmen

Jacob Perrenet, TU/e Informatica
Jan Friso Groote, TU/e Informatica
Eric Kaasenbrood, TU/e Informatica

Samenvatting
Onderzoek naar informaticaonderwijs staat nog in de kinderschoenen. Het kan voortbouwen op de traditie bij het wiskundeonderwijs. Ons onderzoek, uitgevoerd bij de Bacheloropleiding aan de Technische Universiteit Eindhoven, vindt zijn inspiratie in theorieën over wiskundige denkniveaus. Vier abstractieniveaus werden verondersteld in het denken over algoritmen. Een vragenlijst werd geconstrueerd om bij verschillende groepen studenten het beantwoordingsniveau te meten. De resultaten toonden de aanwezigheid van niveauverschillen. Tussen opvolgende jaargroepen trad niveauverhoging op en er was groei in de loop van een jaar. Samenhang tussen gemeten denkniveau en behaald tentamencijfer bij algoritmevakken was er echter nauwelijks. Ook docenten vulden de vragenlijst in.
Keywords: algoritmen, abstractie, curriculum, didactisch onderzoek

Informaticaonderwijs en onderwijsonderzoek aan het begin

Franquinet (2004) schetst de gang van het informaticaonderwijs naar de officiële status in het vo-curriculum. Zo ver is het nog niet. Pas wanneer een vak een volwassen plaats heeft veroverd in primair of secundair onderwijs komt onderzoek naar het onderwijzen en leren van dat vak goed op gang. Ook wereldwijd is er nog lang geen traditie, zoals bij het wiskundeonderwijs, al was het maar, omdat informatica nog zo'n jonge wetenschap is. Net zoals de informatica door de wiskunde is beïnvloed, kan ook het onderzoek van het informaticaonderwijs zich laten inspireren door wat binnen de wereld van het wiskundeonderwijs reeds tot stand is gebracht (Almstrum, e.a., 2002).

Niveautheorie

Bij niveautheorie denken we in Nederland in de eerste plaats aan het werk van Van Hiele (1986). Internationaal zo mogelijk nog bekender is het verwante werk van Skemp (zie bijvoorbeeld Tall en Thomas, 2002). Israëlische onderzoekers van het informaticaonderwijs, zoals Hazzan (2002) en Aharoni (2000) lieten zich door Skemp en zijn collega's inspireren, op zoek naar denkniveaus binnen de informatica. Ze hielden zich bezig met begrippen als berekenbaarheid en datastructuur. De mate van abstractie van een denkniveau heeft volgens hen meerdere aspecten, maar het volgens ons meest fundamentele is abstractieniveau als uitdrukking van de verhouding tussen proces en object. In deze opvatting bereikt een student een hoger abstractieniveau met betrekking tot een bepaald concept, wanneer hij of zij processen en relaties tussen objecten die verband houden met dat concept, weer als een nieuw soort objecten kan zien. We werken dit uit voor het begrip algoritme.

Niveaus bij het begrip algoritme

Toegepast op het begrip algoritme is de volgende door ons gehanteerde vierdeling in niveaus mogelijk:

1. Het executieniveau: het algoritme is een specifieke run op een concrete specifieke machine; de benodigde tijd voor uitvoering wordt door die machine bepaald.
2. Het programmaniveau: het algoritme is een proces, beschreven door een specifieke, uitvoerbare programmeertaal; de uitvoeringstijd hangt af van de input.
3. Het objectniveau: het algoritme staat los van een specifieke programmeertaal; het wordt niet meer als proces, maar als object gezien; bij de constructie van een algoritme worden datastructuur en invariantie-eigenschappen gebruikt; meta-eigenschappen, zoals terminatie, zijn relevant en de benodigde tijd wordt beschouwd in termen van ordegrootte als functie van de input.
4. Het probleemniveau: het algoritme kan als een object worden gezien met de eigenschappen van een 'black box'; het perspectief is geworden: gegeven een probleem, welk type algoritme is geschikt? Problemen kunnen worden gecategoriseerd volgens geschikte algoritmen; een probleem heeft een intrinsieke complexiteit.

Geen vaste niveaus

Hazzan (2002) en Aharoni (2000) onderzoeken niveaureductie. Door interviews met studenten en analyse van uitwerkingen van opgaven ontdekten ze dat studenten vaak de neiging hebben problemen op een lager abstractieniveau aan te pakken dan eigenlijk bedoeld was. Onze benadering is iets anders. We veronderstellen dat er verschillende niveaus mogelijk zijn en dat studenten in de loop van hun studie groeien in hun niveau. Bij het bereiken van een hoger niveau gaat het oude niveau niet verloren, maar wordt opgenomen in het nieuwe geheel. Een probleem kan een bepaald niveau van denken oproepen. Een student heeft wel de mogelijkheid maar niet de noodzaak om op het hoogst beschikbare niveau te reageren.

De Eindhovense opleiding

Het algoritmeonderwijs heeft een apart karakter bij de opleiding Technische Informatica aan de Technische Universiteit Eindhoven. Lange tijd werd het curriculum gedomineerd door de traditie van de zogenaamde Eindhovense School met een fundamentele aanpak van het algoritmeonderwijs, gekenmerkt door gebruik van de wiskunde als effectief ontwerpgereedschap, door correctheid van algoritmen via constructie in plaats van verificatie achteraf en door systematisch gebruik van abstractie om complexiteit te beheersen. Recent werd de moderne ontwikkeling van software engineering, met constructie van systeemcomponenten gevolgd door het samenstellen daarvan tot grotere gehelen, in het curriculum opgenomen. Hoewel in discussie wordt de grondige methode van correctheid via constructie bij het algoritme onderwijs nog steeds gebruikt. De keuze voor het begrip algoritme als onderzoeksobject is dus een voordehandliggende, gezien het karakter van de opleiding.

Hypothesen

- We verwachten niveaus te kunnen onderscheiden binnen de groep van bachelorstudenten.
- We verwachten dat de groep studenten in het derde jaar een hoger niveau heeft dan de groep in het tweede jaar, en die weer hoger dan de groep in het eerste jaar.
- We verwachten niveauverhoging bij een zelfde groep studenten gedurende het jaar.

- We verwachten dat het niveau van een student aan het eind van een onderwijsperiode in een algoritmevak samenhangt met de mate van succes op het tentamen over dat vak.

Tenslotte onderzoeken we of de onderwijsgevenden ook enig globaal zicht hebben op het denkniveau van hun studenten.

Opzet

Eerst is een vragenlijst geconstrueerd en een bijbehorend scoringssysteem ontwikkeld. De vragenlijst werd binnen het academisch jaar 2003-2004 afgenomen bij de tentamens van vijf algoritmegerelateerde vakken, te weten:

- Programmarealisatie 1, op het eind van het eerste trimester van het eerste jaar (1.1.)
- Ontwerp van Algoritmen 1, op het eind van het tweede trimester van het eerste jaar (1.2)
- Ontwerp van Algoritmen 2, op het eind van het eerste trimester van het tweede jaar (2.1)
- Ontwerp van Algoritmen 3, op het eind van het derde trimester van het tweede jaar (2.3.)
- Complexiteit, op het eind van het eerste trimester van het derde jaar (3.1.).

Daarnaast werd de vragenlijst voorgelegd aan een aantal van de betrokken onderwijsgevenden, enerzijds bij collegedocenten en anderzijds bij de praktische oefeningen betrokken instructeurs en student-assistenten.

De vragenlijst

De uiteindelijke vragenlijst bestaat uit zeven items. De lijst begint met het volgende item:

0. Geef je definitie van algoritme.

Dit wordt gevolgd door zes items in de vorm van een vraag of men het eens is met een bepaalde uitspraak (eens, oneens, eens en oneens kan beide, weet niet) plus de vraag om argumenten bij het gekozen alternatief.

1. Een algoritme is een programma, geschreven in een programmeertaal.
2. Twee verschillende programma's in dezelfde programmeertaal kunnen implementaties zijn van hetzelfde algoritme.
3. De correctheid van een algoritme is in het algemeen te bewijzen door de implementatie te testen met slim gekozen testcases.
4. Een geschikte maat om te meten hoe lang een bepaald algoritme erover doet om een bepaald probleem op te lossen, is de tijd die het kost in milliseconden.
5. De complexiteit van een probleem is onafhankelijk van de keuze van het algoritme waarmee je het oplost.
6. Bij ieder probleem is het mogelijk dat in de toekomst algoritmen worden gevonden die een ordegrootte efficiënter zijn dan de nu bekende.

Bij de vragenlijst werden gedetailleerde scoringsregels ontwikkeld om per vraag het niveau van het antwoord te meten (1, 2 of 3 bij de items 0 tot en met 4; 1, 2, 3 of 4 bij de items 5 en 6). Oorspronkelijk was de lijst langer, maar met deze verkorte lijst lukte het twee scoorders tot voldoende overeenstemming te brengen (Cohens Kappa .64). Uit de serie gescoorde antwoorden van een student werd vervolgens het gemiddelde antwoordniveau bepaald als de mediaan van de gescoorde abstractieniveaus.

Resultaten tussen jaargangen

Een aantal studenten gaf geen of onduidelijke argumentatie bij hun antwoorden; bij

ongeveer 85% bleek een gemiddeld antwoordniveau wel te bepalen. Het bleek inderdaad mogelijk verschillende niveaus te onderscheiden, zij het beperkt. In tabel 1 worden de groepen vergeleken aan het eind van respectievelijk trimester 1.1, 2.1 en 3.1. Vrijwel alle studenten hebben een gemiddeld antwoordniveau van 2 tot 3.

Tabel 1: Abstractieniveau bij verschillende jaargroepen

Jaargroep in trimester	Percentage studenten met niveauscore				Aantal
Niveauscore	1.5	2	2.5	3	
Jaargroep 1 eind 1.1	4	50	6	40	67
Jaargroep 2 eind 2.1		21	7	72	58
Jaargroep 3 eind 3.1		8	2	90	72

In hogere jaren is het antwoordniveau inderdaad in het algemeen hoger (een significante rangcorrelatie Spearmanns Rho van 0.48).

Resultaten door een jaar heen

Volgens verwachting groeide het gemiddelde antwoordniveau in de loop van een jaar bij de meeste studenten. In tabel 2 zijn de resultaten van de twee eerstejaars metingen en de twee tweedejaars metingen weergegeven. De percentages zijn gegeven van de studenten met respectievelijk een hoger, zelfde en lager gemiddeld antwoordniveau.

Tabel 2: Niveaugroei gedurende een jaar

Jaargroep (trimesters)	% met hoger niveau	% met zelfde niveau	% met lager niveau	Aantal
1 (1.1→1.2)	48	44	8	36
2 (2.1→2.3)	34	56	10	48

In beide gevallen is er sprake van significante niveauverhoging (Wilcoxon Signed Ranks Test met Z=-2.47 en -2.27 respectievelijk).

Resultaten samenhang met tentamen cijfers

Slechts bij één tentamen was er een kleine maar significante correlatie tussen gemiddeld antwoordniveau bij de afsluiting van een vak en het tentamencijfer voor het vak en wel bij het derdejaars vak Complexiteit. Bij de andere vakken was de correlatie telkens dicht bij 0. Zie tabel 3.

Tabel 3: Correlatie tussen abstractieniveau en tentamencijfer bij de betrokken vakken

Vaktentamen	**Rangcorrelatie**	**Aantal**
Programmarealisatie 1	.14	58
Ontwerp van Algoritmen 1	.05	63
Ontwerp van Algoritmen 2	-.05	44
Ontwerp van Algoritmen 3	.09	72
Complexiteit	.27*	72
* = significant op 0.05 (tweezijdig)		

Resultaten docenten

Ruim de helft van de onderwijsgevenden betrokken bij de uitgekozen algoritmevakken, collegedocenten, instructeurs en student-assistenten, vulde ook de vragenlijst in. Dit deden ze twee keer: vanuit het perspectief van de gemiddelde student die voldoende de stof beheerste om voor het tentamen te slagen en vanuit het perspectief van de student die onvoldoende de stof beheerste. Ook de docenten scoorden op deze wijze van niveau 2 tot niveau 3 en ze scoorden als goede student een beter niveau dan als zwakke student. Er was echter geen samenhang van de hoogte van het niveau met de jaargroep en er was ook geen niveauverhoging zichtbaar bij dezelfde groep in de loop van het jaar, zoals valt af te lezen uit tabel 4. Verder gaven vooral de collegedocenten aan het een moeilijke taak te vinden. Soms namen ze voor de goede student zichzelf en konden ze voor de zwakke student helemaal geen antwoord geven.

Tabel 4: Anwoordniveaus volgens docenten

Trimester	Onderwijsrol	Geschat niveau studenten	
		'Zakker'	'Slager'
1.1	SA	3	3
	SA	2	3
	SA	2	2
	CD	-	3
1.2	CD	-	2
	I	2	2.5
	I	2	3
	I	2	3
2.1	I	2	3
	CD	-	3
2.3	I	2	2.5
	I	-	3
3.1	CD	2	3
SA = studentassistent; CD = collegedocent; I = instructeur; - = missend of niveau niet te bepalen			

Conclusies en Discussie

Het is gelukt een vragenlijst te construeren om t.a.v. het begrip algoritme het (gemiddeld) denkniveau van studenten te meten. De betrouwbaarheid – hoe goed gemeten wordt wat gemeten wordt – hebben we op geaccepteerd niveau kunnen brengen. De validiteit – in hoeverre gemeten wordt wat bedoeld was te meten – zou nog versterkt kunnen worden. We analyseerden bij een groot aantal studenten de argumentatie bij de gekozen antwoorden van de vragenlijst. We zouden in vervolgonderzoek ook nog bij een kleiner aantal studenten zowel de vragenlijst als een langer mondeling interview kunnen afnemen om hun niveau van algoritmebegrip te peilen. Wanneer we duidelijke verbanden zichtbaar zouden kunnen zichtbaar maken tussen het niveau volgens de vragenlijst en het niveau volgens het interview, zou de validiteit daarmee nog versterkt zijn.

Vrijwel alle studenten kwamen bij hun gemiddelde beantwoording van de vragen in de range niveau 2 tot niveau 3 terecht. Mogelijk is het eerste niveau sterker aan het eind van het VWO te vinden of aan het begin van het HBO; voor een betere zichtbaarheid van het vierde niveau zou de vragenlijst moeten worden uitgebreid en zou bij masterstudenten gemeten moeten worden. Het zou ook interessant zijn de vragenlijst aan studenten van een andere universitaire informaticaopleiding voor te leggen.

Er bleek een hoger niveau te zijn in opvolgende jaargangen en er was niveauverhoging te constateren gedurende het jaar. Aangezien het gaat om correlaties moeten we voorzichtig zijn met het trekken van conclusies aangaande dat het algoritmeonderwijs ook de oorzaak zou zijn van deze niveauverhoging. Voor dergelijke conclusies zou experimenteel onderzoek nodig zijn: een opzet met vergelijkbare groepen, die zo mogelijk alleen zouden verschillen op het punt van het al of niet volgen van de betreffende vakken. Dit is praktisch lastig uitvoerbaar. De gevonden resultaten kunnen ook door andere factoren veroorzaakt zijn, bijvoorbeeld het onderwijs in het algemeen of zelfs het ouder worden op zich.

Er was onverwacht weinig samenhang tussen het antwoordniveau van een student voor onze vragenlijst en het antwoordniveau op het algoritmetentamen tezelfdertijd. Misschien hadden die docenten gelijk die vooraf al aangaven dat het bij hun vak (onderwijs in de zogenaamde Guarded Command Language voor het correct construeren van elementaire algoritmen, zie Kaldewaij, 1990) eigenlijk meer om precisie gaat dan om abstractie. Het zou ook interessant zijn cijfers voor andere vakken uit dezelfde tentamenperiode in de vergelijking te betrekken.

Bovenstaand gebrek aan samenhang tussen tentamencijfer en niveau verklaart ook deels de moeilijkheid van docenten de vragenlijst in te vullen vanuit het perspectief van de zwakke en de sterke student. Aan de andere kant was het opvallend hoe lastig docenten het vonden zich in de denkwereld van de student – vooral de zwakke student – te verplaatsen. Zonder enig zicht daarop bestaat het risico, dat - zeker bij colleges met weinig interactie - het gegeven verhaal geen aansluiting vindt. Bij de instructies is door de meer intensieve interactie de kans op aansluiting met de denkwereld van de student groter. Het zicht op die denkwereld is aan de ene kant te vergroten in de praktijk: door onderwijs met veel interactie; aan de andere kant door onderzoek, bijvoorbeeld zoals hier beschreven. Een verbreding zou zijn het identificeren van *alle* belangrijke begrippen in de informatica en het karakteriseren van de ontwikkeling die studenten daarin doormaken.

In Nederland kennen we het Utrechtse Freudenthal Instituut voor onderzoek van het wiskundeonderwijs vanuit de vakdidactiek en ook Groningen en sinds kort Heerlen moeten op dit punt niet vergeten worden; daarnaast is er onderzoek van het wiskundeonderwijs vanuit onderwijskundige en psychologische context. Meestal betreft het dan primair en secundair onderwijs, maar ook het hoger onderwijs komt langzamerhand meer binnen het gezichtsveld. Wanneer nu eindelijk het informaticaonderwijs vaste voet op HAVO en VWO gaat krijgen en er volwaardige lerarenopleidingen informatica komen, lijkt ook het moment gerechtvaardig voor het opzetten van een Instituut voor Onderzoek en Ontwikkeling van het InformaticaOnderwijs[1]. Zo'n instituut zou kunnen profiteren van onderzoek en ontwikkeling van het wiskundeonderwijs, zoals bepleit door Almstrum e.a. (2002), van onderzoek en ontwikkeling bij andere exacte vakken (Almstrum e.a., 2003), van relevant onderzoek uit de sociale wetenschappen, en last but not least van de praktische ervaringen vanuit het CODI (Consortium Omscholing Docenten Informatica) en vanuit de informaticadocenten zelf.

Referenties

- Aharoni, D. (2000), Cogito, Ergo, Sum! Cognitive Processes of Students Dealing with Data Structures. *Proceedings SIGCSE,* Austin, blz. 26-30.
- Franquinet, R. (2004), Informatica VO uit de kinderschoenen; *Tijdschrift voor informaticaonderwijs, 13e jaargang, nummer 2*, blz. 46-49.
- Almstrum, V.L. e.a. *(*2002), Import and Export to/from Computing Science Education: The Case of Mathematics Education Research. *Proceedings ITiCSE,* Aarhus, blz. 193-194.
- Almstrum, V.L., e.a. (2003), Transfer to/from Computing Science Education: The Case of Science Education Research. *Proceedings SIGCSE,* Reno, blz. 303-304.
- Hazzan, O. Reducing Abstraction Level when Learning Computability Concepts. *Proceedings ITiCSE,* Aarhus*,* 2002, blz. 156-160.
- Kaldewaij, A. *Programming: The Derivation of Algorithms.* Prentice Hall International, UK, 1990.
- Tall, E. & Thomas, T. (Ed.) (2002), *Intelligence, Learning and Understanding in Mathematics; a tribute to Richard Skemp.* Post Pressed, Flaxton.

Noot

1. Alsof het was afgesproken: op het NIOC-congres werd inderdaad de eerste stap gezet om tot een IOIO te komen.

Nooit meer met de rug naar de klas?
Analyse van onderwijsprocessen in een adaptieve virtuele leeromgeving

Frans Mofers, Harrie Passier,
Open Universiteit Nederland, faculteit Informatica

Samenvatting
De kwaliteitszorg rond onderwijsprocessen in elektronische leeromgevingen staat nog in zijn kinderschoenen. Bestaande benaderingen om de participatie van studenten in het onderwijs te beoordelen, resultaten te evalueren en de effectiviteit en efficiëntie te beoordelen, voldoen in dit soort omgevingen vaak niet. Nieuwe benaderingen moeten hiervoor ontwikkeld worden. In dit artikel worden de resultaten beschreven van een onderzoek naar het toepassen van audit principes in een elektronische leeromgeving.

Sleutelwoorden: audit, feedback, elektronische leeromgeving, kwaliteitszorg

Inleiding
Virtuele leeromgevingen worden steeds meer ingezet in diverse onderwijssituaties binnen het Hoger Onderwijs. De klassieke leeromgeving met de docent voor het bord of gebogen over de overhead projector wordt aangevuld of vervangen door een Elektronische Leeromgeving (ELO). Het inzetten van ELO's wordt verder versterkt door nieuwe uitdagingen waarvoor het HO zich gesteld ziet zoals: heterogene instroom, vraaggestuurd onderwijs en levenslang leren (Geloven, 2004).
De ELO wordt niet alleen meer gevuld met informatie over het onderwijsproces maar ook de onderwijscontent, bijvoorbeeld in de vorm van leerobjecten, krijgt een prominente plaats (zie bijvoorbeeld Altenhofen, 2002). De leeromgeving wordt daarnaast ook ingezet om groepswerk te ondersteunen, zoals het uitwisselen van documenten, het discussiëren over het onderwijs tussen studenten onderling en tussen studenten en docenten en het uitvoeren van peer reviews (Soller, 1999).
Het beoordelen of de leeromgeving, de processen en presentatie, en het daarin opgenomen materiaal effectief gebruikt wordt en aansluit bij de wensen van studenten en docenten is daarbij momenteel problematisch: de docent mist het directe contact als één van de instrumenten bij de beoordeling of het onderwijs aan zijn doelstellingen en de doelstellingen van de student voldoet. Inzicht in hoeverre deze doelstellingen wordt bereikt is echter wel essentieel (Murray, 1999).

Voorbeelden gebruik van leermaterialen
Een tweetal voorbeelden waarin de uitwisseling van leermateriaal op brede schaal wordt beoogd binnen het Nederlandse Hoger Onderwijs zijn Espelon en het LOK-project (zie bij Informatie op internet). In Espelon wordt zowel digitaal onderwijsmateriaal aangeboden maar ook samenwerking georganiseerd via fora etc. In het kader van het LOK-project wordt door een groot aantal HO-instellingen een leeromgeving gedeeld met taken op het gebied van kennistechnologie. Het verkrijgen van kwantitatieve maar ook kwalitatieve feedback over het gebruik van de leermiddelen is in dit soort uitwisselingssituaties moeizaam.

Ontwikkelingen vanuit R&D
Vanuit R&D gaat de aandacht in sterke mate uit naar standaardisatie waardoor content op eenvoudige wijze uitgewisseld kan worden. In navolging van deze standaardisatietendens op het gebied van content wordt ook gezocht naar mogelijkheden om de communicatieprocessen en de logistieke ondersteuning te standaardiseren (Santos, 2004).
Deze voorbeelden voor gebruik en distributie van leermaterialen illustreren de virtualisering van het onderwijs. Tevens tonen deze voorbeelden de noodzaak aan om in het materiaal voorzieningen in te bouwen waarmee aan de kwaliteitszorg invulling gegeven kan worden, ook wanneer het materiaal op een andere plek als bij de ontwikkelaar gebruikt wordt.

Gevolgen virtualisering van het onderwijs
Wanneer het onderwijs in belangrijke mate plaatsvindt in een elektronische leeromgeving bestaat de kans dat de afstand tussen de onderwijsgever en de afnemer van het onderwijs ongewenst groot wordt. Dit kan nadelige gevolgen hebben voor de mogelijkheden om zicht te houden op de kwaliteit van het onderwijs. Juist nu structurele aandacht voor kwaliteitszorg sterk in belang toeneemt, is dit een ongewenst neveneffect van de ontwikkeling waarbij steeds vaker een virtuele omgeving wordt ingericht om onderwijstaken aan te bieden. Daartegenover staat echter dat in deze nieuwe infrastructuur ook nieuwe mogelijkheden ontstaan om zicht te houden op leerprocessen die zich afspelen in een dergelijke leeromgeving (Rossett, 2002). Er hoeft daarbij niet vervallen te worden in big-brother-achtige informatievergaring. De informatie kan toegespitst worden op de behoefte van de onderwijsverzorger: zoveel informatie als nodig is voor verbetering van de processen en de inhoud, op het tijdstip waarop de informatie nodig is.
Het onderzoek waar in deze paper op ingegaan wordt richt zich erop relevante en toegespitste informatie te vergaren voor de cursusontwikkelaar, de begeleider en eventueel ook voor de manager van de onderwijsorganisatie over het gebruik van het cursusmateriaal en de leerresultaten die behaald worden. De informatieverschaffing naar de student over voortgang en resultaten valt buiten het kader van dit onderzoek.

Audit rapportages
Door gebruik te maken van een in andere domeinen bekende benadering als *auditing,* kan op het juiste niveau inzicht verkregen worden dat direct kan leiden tot bijstellingen van de didactiek en/of inhoud van een onderwijsmodule.
De afgelopen jaren is veel vooruitgang geboekt bij de ontwikkeling van op XML gebaseerde representatiemechanismen als EML (Hermans, 2004) waarmee het onderwijskundig ontwerp en de leermaterialen kunnen worden vastgelegd in een gestandaardiseerde vorm. Hierdoor ontstaat een vrijheid van keuze voor een afspeelomgeving en ook hergebruik van materiaal is beter mogelijk.
Ook wordt het nu mogelijk om volledig nieuwe vormen van informatieopslag en informatie-uitwisseling te ontwikkelen. Het is bijvoorbeeld mogelijk om samen met het onderwijskundig ontwerp, ook informatie over de doelstelling die de organisatie heeft met dit materiaal in de vorm van kritieke succesfactoren en procesindicatoren eenduidig vast te leggen. Het is vervolgens noodzakelijk om uit de leeromgeving toegespitste informatie te vergaren over het gebruik van de leermaterialen. De informatie die op deze manier beschikbaar komt, zowel vanuit het ontwerp alsook vanuit het daadwerkelijk gebruik, wordt geanalyseerd en in rapporten vastgelegd. Auteurs en docenten kunnen deze informatie gebruiken om het onderwijsmateriaal te verbeteren.

De rapportages die voortkomen uit een audit van een elektronische onderwijsleeromgeving kunnen zowel gericht zijn op:
- efficiency: om te vermijden dat modules ontwikkeld en aangeboden worden die didactisch en inhoudelijk onvoldoende toegevoegde waarde hebben in termen van gebruik en slaagpercentage
- effectiviteit: levert bijvoorbeeld elke module voldoende leerrendement op en is de studeerbaarheid voldoende;
- algemene doelstellingen van de instelling: worden de doelstellingen gerealiseerd in de elektronische leeromgeving in termen van bijvoorbeeld studielast, slaagpercentages of percentage afvallers.

Adaptieve virtuele leeromgeving
De context waarbinnen dit onderzoek zich afspeelt is het Alfanet (Active Learning For Adaptive Internet, zie bij Informatie op internet) project. Binnen Alfanet wordt een adaptieve virtuele leeromgeving ontwikkeld. De leeromgeving biedt adaptiviteit op studentniveau. Op basis van studentkenmerken, afgelegd studiepad en persoonlijke voorkeuren oefent de student invloed uit op de leertaken die gepresenteerd worden, de route door de taken alsmede de vorm waarin de taken gepresenteerd worden. Ook zijn hulpmiddelen aanwezig voor collaboratief leren (Soller, 1999) waardoor impliciet adaptiviteit aanwezig is in de vorm van samenwerkend leren.
Deze op de student gerichte adaptiviteit wordt aangevuld met aanvullende adaptiviteit die door de ontwikkelaar van leertaken aangebracht kan worden. De ontwikkelaar kan een leertaak aanpassen bijvoorbeeld op basis van bevindingen uit het leerproces of op basis van actuele informatie uit de buitenwereld.

Ontwerpstandaarden
Deze verschillende vormen van adaptiviteit worden in belangrijke mate mogelijk gemaakt doordat in het Alfanet project gebruik gemaakt wordt van standaarden waarmee leerobjecten gespecificeerd worden. Centraal staat de IMS-LD-standaard (Learning Design, gebaseerd op EML: Educational Modelling Language) (Hermans, 2004) waarmee onderwijsmateriaal en -processen beschreven kunnen worden. Specifiek voor het ontwikkelen en aanbieden van toetsen is gebruik gemaakt van de IMS-QTI-standaard (Question and Test Interoperability). Daarnaast wordt de IEEE-LOM-standaard (Learning Objects Metadata) gehanteerd waarmee meta-informatie toegevoegd kan worden aan het ontwerp. Het materiaal dat is ontwikkeld met behulp van specifieke editors die deze drie standaarden ondersteunen, wordt afgespeeld in een webplayer waarmee de cursus via het web aangeboden kan worden. De audit service maakt gebruik van specifieke traces die door de player gegenereerd worden. Na analyse kunnen rapporten aangemaakt worden die enerzijds gebaseerd zijn op informatie die vastgelegd is in het ontwerp (veelal de normen) en anderzijds informatie die afkomstig is uit de traces (de gemeten waarden)

Indeling van deze paper
Allereerst gaan wij in op het Alfanet project waarbinnen de adaptieve leeromgeving, een aantal specifieke onderwijsmodellen, alsmede de audit module ontwikkeld zijn. Vervolgens gaan wij in op de principes van auditing en het toepassen daarvan bij het faciliteren van de kwaliteitszorg in virtuele onderwijsprocessen. Tenslotte zullen wij de eerste resultaten bespreken en ingaan op ons toekomstig onderzoek.

Een adaptieve elektronische leeromgeving

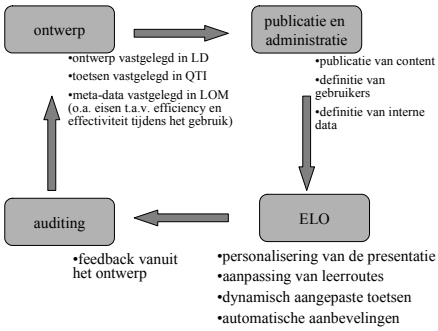
Het Alfanet project
"Alfanet concentrates on the emerging market of e-learning, an area that will take advantage

of the new technologies related to the internet, human interaction, and machine learning. More specifically, Alfanet will allow:
– Individuals to have interactive, adaptive and personalised learning through the internet.
– Service providers or educational centres to provide e-learning services adapted to the individual learners background, experience and behaviour.
– Learning content providers to produce learning contents adapted to the personal needs of the individual learner." (zie bij Informatie op internet).

Voor de audit functie zijn specifiek de volgende aspecten van belang:
– er wordt gewerkt vanuit procesoptiek;
– er wordt gewerkt vanuit een onderwijskundig model;
– het onderwijskundig model wordt expliciet vastgelegd in een gestandaardiseerde taal.

Het onderwijsproces
Zoals bij de meeste vormen van onderwijs is ook bij een virtuele leeromgeving het expliciteren van het onderwijskundig model en het onderwijskundig proces een hulpmiddel voor een succesvol en beheersbaar leerproces.
Het onderwijskundig proces gaat uit van de processtappen: analyse, ontwerp, implementatie en evaluatie. Wanneer deze processtappen doorlopen zijn, wordt vanuit de analyse een aangepast ontwerp gemaakt. In figuur 1 is deze algemene cyclus expliciet uitgewerkt voor de ontwikkelcyclus binnen Alfanet, waarbij gebruik gemaakt wordt van open standaarden en hulpmiddelen die hierop afgestemd zijn.

In de ontwerpfase wordt het lesmateriaal vanuit een didactisch model in gestandaardiseerde documenten vastgelegd met behulp van specifieke editors. Niet alleen het didactisch model en het cursusmateriaal worden in leerobjecten elektronisch opgeslagen, maar ook de eisen ten aanzien van de effectiviteit en efficiency bij het gebruik. Deze documenten worden vervolgens aangeboden voor publicatie in een leeromgeving. Tenslotte brengt auditing de gebruiksgegevens en de doelstellingen uit het ontwerp met elkaar in relatie en presenteert de analyse hiervan via auditrapporten. Deze vormen vervolgens weer de invoer voor een volgende ontwerp- en aanbiedingscyclus.



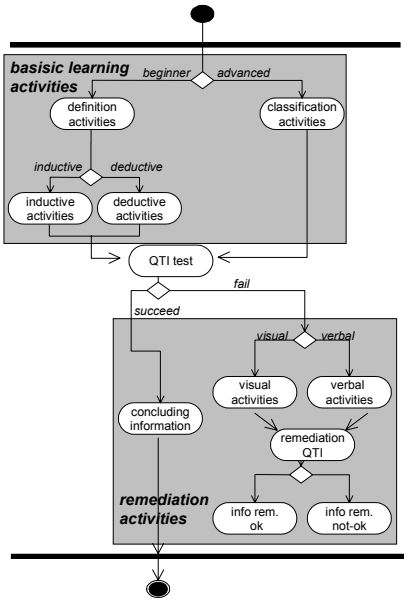
Figuur 1 ontwikkelcyclus in het Alfanet project

Het onderwijskundig model
In principe kan een dergelijke adaptieve omgeving vele onderwijskundige modellen ondersteunen. In het kader van het Alfanet project is een basaal didactisch model gekozen dat als template fungeert bij de ontwikkeling van een aantal cursussen. Dit model is ontwikkeld met het oog op de gewenste adaptieve leerroutes door het materiaal die dit model mogelijk maakt.

Een cursus is opgedeeld uit leertaken. Elke leertaak is opgebouwd uit activiteiten die conform figuur 2 in het didactisch model geplaatst zijn.

Allereerst is de leertaak opgebouwd uit een verzameling basismodules en modules die een remediërende functie hebben. In de basismodule worden definitieactiviteiten (het leren

van concepten) en classificerende activiteiten (het kunnen werken met concepten) onderscheiden.



Figuur 2 didactisch model van een leertaak

Het pad door de activiteiten dat door een individuele student in de verschillende modules gevolgd wordt, is allereerst afhankelijk van een aantal persoonlijke eigenschappen:
– de startcompetentie (beginner of gevorderd);
– de leerstijl (inductief of deductief);
– de cognitieve voorkeur (visueel of verbaal).
Het pad is vervolgens afhankelijk van toetsresultaten die in de leertaken zijn opgenomen.

Aanvullend aan deze routes die in het ontwerp vastgelegd zijn, kan de student op elk ogenblik elke leermodule kiezen die binnen een leertaak beschikbaar is. Als derde adaptief element is er een adaptieve module beschikbaar die voorstellen over het te volgen leerpad presenteert aan de student op basis van een analyse van het studeergedrag van alle studenten die eerder deze leertaak doorlopen hebben.

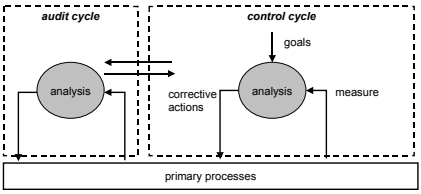
Standaarden
Het onderwijskundig model is geïmplementeerd in de vorm van een template die per cursus en per leertaak gevuld kan worden met concrete activiteiten. Het template is gespecificeerd in IMS-LD.
Een tweede standaard die gehanteerd wordt in de Alfanet leeromgeving is QTI. Deze standaard biedt de mogelijkheid om ook toetsen op een gestandaardiseerde manier vast te leggen.
Een derde standaard die een rol speelt in het Alfanet systeem is de LOM-standaard. Deze standaard wordt gebruikt om: 1) in de ontwerpfase informatie op metaniveau vast te leggen die later gebruikt wordt om de analyse van het leerproces te sturen en 2) informatie vast te leggen over de activiteiten zoals geraamde studietijd voor een activiteit of moeilijkheidsgraad van een activiteit (normen).

Samen zorgen deze drie standaarden ervoor dat het onderwijskundig model eenduidig beschreven wordt en een geautomatiseerde analyse van het ontwerp door de audit service in het Alfanet systeem mogelijk is.

Audit en het onderwijsproces

Audit in een bedrijfskundige context
Het instituut voor Internal Aditors Nederland definieert het begrip Internal Auditing als volgt: “Internal Auditing is een onafhankelijke, objectieve "assurance" (geven van zekerheid) en consulting (geven van advies) activiteit met de bedoeling waarde toe te voegen aan en verbetering te brengen in de operaties van een organisatie. Internal Auditing helpt een organisatie om haar doelen te verwezenlijken door een methodische, ordelijke bena-

dering om de effectiviteit van risicomanagement, controle en beheersingsprocessen te evalueren en verbeteren.".
Auditing is gebaseerd op het algemene feedback principe waarbij een proces wordt gestuurd door de procesoutput te meten en te vergelijken met normen. Het verschil tussen gemeten waarde en norm vormt de basis voor een analyse en eventuele bijsturing (Distefano, 1988).
In figuur 3 is weergegeven hoe audit gezien kan worden als een stuurproces dat aanvullend aan de primaire sturing van bedrijfsprocessen (control) een tweede niveau van sturing toevoegt. Naast het primaire proces wordt ook het controlproces door de audit functie geobserveerd en gebruikt voor een tweede-orde bijsturing van de primaire processen (Boer, 2002).

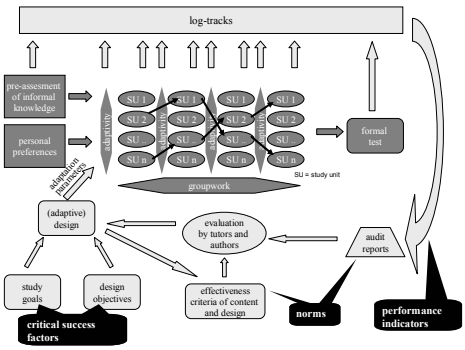


Figuur 3 basismodel audit

Er dient vermeden te worden dat de auditsturing doublures vertoont ten opzichte van de primaire sturing en het gehele regelmechanisme te complex wordt. Daarom worden voor deze tweede-orde sturing specifieke parameters gehanteerd.
Allereerst worden de doelstellingen van het onderwijssysteem vastgelegd in kritieke succesfactoren. Vervolgens worden deze kritieke succesfactoren concreet vertaald naar normen waaraan het systeem dient te voldoen en bijbehorende meetpunten of procesindicatoren.
In de volgende paragraaf geven wij aan op welke manier deze audit-sturing geconcretiseerd wordt in de onderwijscontext.

Een model voor het toepassen van audit in het onderwijsproces

In figuur 4 is schematisch de (levens-) cyclus beschreven bij het doorlopen van een cursus in een adaptieve leeromgeving zoals in het Alfanet project gerealiseerd is.



Figuur 4 audit proces in een elektronische leeromgeving

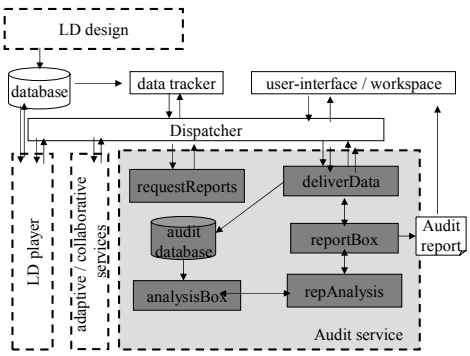
Wij beschrijven dit proces hier kort en beginnen bij de toetsen die een student voorafgaand aan het bestuderen van de cursus uitvoert. In deze eerste stap worden via een toets de persoonlijke voorkeuren bepaald zoals de leerstijl en de cognitieve voorkeur. Ook wordt de voorkennis bepaald. Op basis van deze informatie kan een aanbevolen leerpad door de activiteiten bepaald waarbij zowel de persoonlijke informatie als de informatie uit het didactisch ontwerp gebruikt worden. De student kan dit pad volgen of kan de voorkeur geven aan een ander pad. Deze keuze kan beïnvloed worden door het advies van de adaptieve module. Ook resultaten van tussentijdse toetsen beïnvloeden het aanbevolen leerpad.
De Alfanet leeromgeving bevat daarnaast een platform waar studenten in groepen kunnen samenwerken en documenten uit kunnen wisselen, zowel onderling als met docenten.

In een trace-bestand wordt vastgelegd welke activiteiten uitgevoerd zijn en welke resultaten studenten behaald hebben bij tests. Uit dit trace-bestand worden de waarden per performance indicator afgeleid. Deze gegevens worden in een analyse vergeleken met de door de cursusontwikkelaar bepaalde normen.
Een analyse van de audit rapporten kan vervolgens door de cursusontwikkelaars gebruikt worden om het cursusontwerp of het cursusmateriaal zelf te verbeteren. Eventueel kunnen de audit rapporten ertoe leiden dat de leerdoelen of de ontwerpdoelstellingen bijgesteld worden. Ook kunnen de effectiviteitscriteria bijgesteld worden op basis van een meer realistische kijk die ontstaat door de analyse op hoog niveau van het gebruik van leerobjecten.

Eerste resultaten

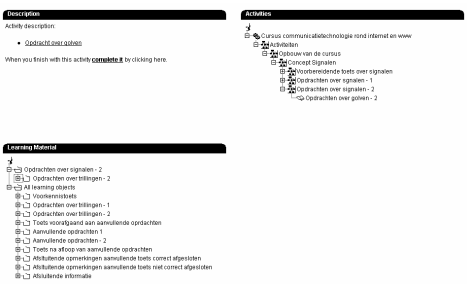
Een eerste prototype van de audit functie is beschikbaar voor de gebruikers van het Alfanet systeem.

Architectuur van het Alfanet systeem
In figuur 5 is de architectuur van de audit service weergegeven binnen de architectuur van het Alfanet systeem.



Figuur 5 systeemopzet van de audit service binnen het Alfanet systeem

De belangrijkste component van het Alfanet systeem vormt de player die het ontwerp kan interpreteren en de leerobjecten aanbiedt aan de student via de 'workspace' in het 'user-interface' (zie ook figuur 6). De dispatcher vormt een centrale communicatiecomponent die zorgt voor de uitwisseling van de gegevens tussen de verschillende componenten. Met de adaptieve en collaboratieve services wordt het leerproces verder ondersteund.
Informatie over het uitvoeren van de activiteiten alsmede de resultaten van de toetsen worden in de database opgeslagen en beschikbaar gesteld aan o.a. de audit service.
In figuur 5 is ook de interne opbouw van de audit service globaal weergegeven. Op basis van een verzoek van bijvoorbeeld een auteur vanuit het gebruikersinterface worden de relevante data voor een audit rapport opgehaald. De data worden opgeslagen in een interne database waarmee snel analyses uitgevoerd kunnen worden. In een aantal stappen wordt vanuit de gegevens in de interne database het rapport aangemaakt en aangeboden aan de aanvrager via het gebruikersinterface.



Figuur 6 Alfanet leeromgeving

In figuur 6 is een deel van de 'workspace' van de Alfanet leeromgeving weergegeven waarbij in het vlak linksboven de naam van de leertaak zichtbaar is waar de student mee bezig is. In het vlak rechtsboven wordt het leerpad weergegeven dat de student heeft doorlopen alsmede de activiteiten die momenteel

geadviseerd worden. Linksbeneden is een overzicht van alle leertaken beschikbaar.
De manier waarop deze overzichten van de activiteiten weergegeven worden kan door de student ingesteld worden hetgeen ook bijdraagt aan de adaptiviteit van de elektronische leeromgeving.

Audit rapporten
Eerste life tests van het Alfanet systeem en de audit service moeten nog plaatsvinden. Wel zijn al audit rapporten beschikbaar op basis van test-runs.
In figuur 7 is (een deel) van het meest basale rapport zichtbaar waarmee een overzicht geboden wordt van de activiteiten die uitgevoerd zijn, door wie deze uitgevoerd zijn, wanneer de activiteit voor het eerst en voor het laatst uitgevoerd is en of de student aangegeven heeft dat deze de activiteit afgesloten heeft.

OVERVIEW OF THE ACTIVITIES FOR A GIVEN COURSE

PARAMETERS:

CourseId

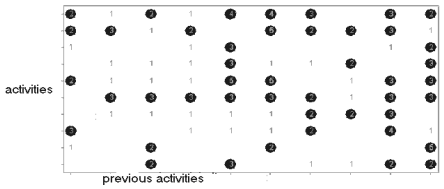
CommtechInformatica

RESULTS:

Learner / activity	First Time	Complete Time	Access Count	Status
ounl1 / RemTest	7-nov-04	7-nov-04	4	compl
ounl1 / Inductive	7-nov-04	7-nov-04	3	
ounl1 / ConceptTest	7-nov-04	7-nov-04	2	compl
ounl1/ Visual	7-nov-04	7-nov-04	1	compl

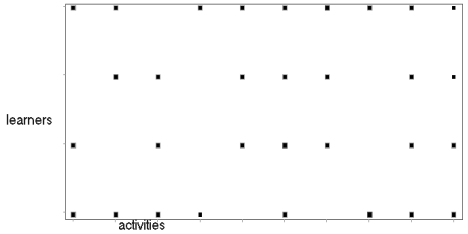
Figuur 7 basaal audit rapport

Ook zijn er rapporten die in meer detail informatie bieden over het leerproces, zoals:
– welke studenten een bepaalde activiteit uitgevoerd hebben;
– de gemiddelde studietijd voor een bepaalde activiteit (performance indicator), afgezet tegen de geraamde studietijd (norm).



Figuur 8 audit rapport dat aangeeft in welke mate activiteiten achtereenvolgens uitgevoerd worden

Ook kan de audit informatie in verschillende grafische vormen weergegeven worden. Deze representatievormen zijn vooral bedoeld om te analyseren of het beoogde leerpad doorlopen wordt dan wel of er voorkeuren bestaan voor specifieke andere leerpaden (zie de figuren 8 en 9).



Figuur 9 audit rapport waarmee het dynamisch gedrag ten aanzien van het achtereenvolgens uitvoeren van activiteiten weergegeven wordt

Toekomstig onderzoek

In vervolgonderzoek willen wij de besproken audit principes valideren aan de hand van het ontwikkelde prototype van een cursus, met studenten die deze cursus bestuderen. Tevens willen we meer complexe analysefuncties gaan ontwikkelen waarmee bijvoorbeeld de

voorkeur voor bepaalde leerpaden geanalyseerd kan worden ten opzichte van de paden zoals deze zijn gedefinieerd en vastgelegd in het ontwerp. Voor deze complexere analyses zijn waarschijnlijk technieken uit de kunstmatige intelligentie nodig (zie o.a. Beck, 1999).

Verantwoording

Het onderzoek is uitgevoerd in het kader van het Alfanet project. Het Alfanet project is gesubsidieerd door de Europese Gemeenschap en valt onder het programma voor Information Society Technologies (IST). De doelstelling van het Alfanet project is het ontwikkelen van nieuwe methoden en services voor actief leren in adaptieve leeromgevingen.

Referenties

Artikelen

Beck J.E. and Stern M. K., 1999, *Bringing back the AI to AI&ED,* Artificial intelligents in education, S.P. Lajoie and M. Vivet (eds), IOS press, Amsterdam

Boer H. den en Zutphen L.C. van, 2002, *Business control en auditing.* Academic Service, Schoonhoven

Distefano J. J., 1988, *Theory and problems of feedback and control systems,* Metrics editions, Schaum's Outlin series

Geloven, M. van et al, 2004. *E-learning trends 2004: standaarden, technologie en eigendomsrecht*. Digitale Universiteit, Nederland

Hermans H. et al, 2004, *Educational modelling language*, in W. Jochems, J. van Merrienboer, & E.J.R. Koper, Integrated eLearning. London: RoutledgeFalmer.

Lindsley O. R., 1999, *From training evaluation to performance tracking,* in Handboek of human performance technology, H. D. Stolovitch and E.J. Keeps (eds), 2nd edition, San Francisco: Jossey-Bass/ Pfeiffer.

Rosmalen P. van et al, 2003, *Towards an open framework for adaptive, agent-supported e-learning,* OUNL, preprint.

Rossett A., 2002, *Walking in the night and thinking about e-Learning*, in The ASTD e-learning handbook – best practices, strategies, and case studies for an emerging field Allison Rossett (eds), MCG.

Santos O. et al*,* 2004, *Authoring a collaborative task extending the IMS-LD to be performed in a standard-based adaptive Learning Management System called aLFanet,* AHCW, 2004.

Soller A. et al, 1999, *Toward Intelligent analysis and support of collaborative learning interaction*, Artificial intelligents in education, S.P. Lajoie and M. Vivet (eds), IOS press, Amsterdam.

Murray T., 1999, *Authoring intelligent tutoring systems: an analysis of the state of the art,* in Artificial intelligence in education, International journal of Artificial Intelligence in Education.

Valcke, H, 2004, *Objecten ontwerpen voor samenwerkend leren*, Informatie, jaargang 46/8, blz. 26-31

Informatie op internet

Alfanet: http://alfanet.ia.uned.es/

LOK: http://www.ntwpracticumnet.ou.nl/lok/

Epselon: http://www.espelon.nl/

Een model voor samenwerkend leren via Internet

Koos Baas, OUNL-Informatica
Frans Mofers, OUNL-Informatica

Samenvatting

Hoge capaciteit internetverbindingen strekken zich nu uit tot in de woningen van studenten. Dit gegeven is in het afstandsonderwijs de katalysator voor de ontwikkeling van allerlei additionele onderwijsmodellen naast de traditionele modellen voor reguliere schriftelijke onderwijs. Deze ontwikkeling beperkt zich niet tot het afstandsonderwijs. Ook in het reguliere onderwijs neemt het 'off campus studeren' hand over hand toe. Het blijkt echter dat het aanbieden van Internetdiensten alleen zoals e-mail, discussiegroepen, webpagina's onvoldoende is om vormen van actief leren mogelijk te maken. Het blijkt zelfs dat kennis hoe dan wél een onderwijsomgeving op Internet moet worden geconstrueerd, nagenoeg ontbreekt. In deze bijdrage beschrijven we een exploratieve aanpak voor de constructie van een dergelijke onderwijsomgeving. Uitgangspunten daarvoor zijn samenwerkend leren op basis van constructivistische principes. Het exploratieve element zit in de tegenstelling die bestaat tussen structurering van het onderwijsproces en samenwerkingsverbanden, en het ontstaan van zelfsturende en zelflerende teams. In de bijdrage beschrijven we een jaar ervaring met deze aanpak en aanbevelingen voor verdere ontwikkeling.

Keywords: Computer Supported Collaborative Learning, CSCL, Constructivisme,

Introductie

Van oudsher zijn instituten voor afstandsonderwijs geïnteresseerd in diverse media en de daarbij horende methoden om de fysieke afstand tussen kennisinstelling en student te verkleinen. Met de komst van het wereldwijde Internet en de verbeteringen daarin qua bandbreedte, fijnmazigheid (Plugge 2003) en de algemeen toegenomen computervaardigheden, is de interesse in en de ambitie om Internet voor afstandsonderwijs te gebruiken alleen nog maar gegroeid. Ook in het contactonderwijs zullen elementen van afstandsonderwijs binnendringen. Niet in het minst door het ontstaan van grote regionale kennisinstituten zal de 'distributie van onderwijs' naar de filialen of naar de studenten thuis, steeds vaker voorkomen. Ook het door de overheid nagestreefde *´zappen´* (BAMA) van studenten tussen onderwijsinstelling naar onderwijsinstelling zal het gebruik van een virtuele leeromgeving verterken. Persoonlijke portfolio´s waarin het studieverleden van de individuele student vastgehouden wordt zullen een belangrijke ondersteunende rol kunnen vervullen.

De vraag die zich vervolgens voordoet is hoe vindt dan die 'distributie van onderwijs' plaats? Is het de 'buitenkant van het onderwijs' met allerlei mededelingen zoals roosters, tentamenuitslagen, opendagen, opleidingen, inschrijfformulieren en postbussen waarin de gemaakte opdrachten elektronisch in kunnen worden gedeponeerd; of gaat het dieper met de ontwikkeling van nieuwe onderwijsvormen op deze 'sea of bytes'? Deze laatste vraag is het onderwerp van dit artikel.

1. Constructivisme

Wanneer het concept van de 'lerende mens' centraal wordt gesteld, komt onvermijdelijke de vraag naar voren van hoe deze mens dan leert? In elk geval gaat het om een mens die zowel in als buiten een kennisinstituut leert. In de regel zullen de grenzen van de aanwezige kennis worden opgezocht en worden gebruikt om nieuwe kennis te vinden, interpreteren en te gebruiken. Met andere woorden, het verkennen van grenzen leidt tot een actief leerproces waarin de student betekenis gaat geven aan nieuwe ontwikkelingen. De opgedane kennis wordt individueel zodanig opgeslagen dat deze later weer kan worden gebruikt (integratie, mentaal model). Wil de 'interne kennis' tot algemene kennis komen, dan moet er worden onderhandeld over het precieze begrip, de duiding van de opgedane kennis en de relatie hiervan met de bestaande kennis. Deze algemene kennisopbouw van begrippen en gedeelde opvattingen komen tot stand door interactie met medestudenten (peers) en docenten. Deze drie elementen, verkenning, interne verwerking en de constructie van algemene kennis zijn de pijlers onder de constructivistisch leren (Alexander 1999). Bij deze opvatting staat leren voorop, niet het onderwijzen. Het is ook aannemelijk dat de kenmerken van constructivisme ook de grondslag vormen voor de praktijk van (onderzoekende) professionals.

Een tweede argument voor de constructivistische aanpak kan afgeleid worden uit het curriculum. Het curriculum kan in termen van de te ontwikkelen professionaliteit de volgende indeling hebben: beginner, gevorderd en semi-professional. De beoogde aanpak appelleert aan het laatste niveau omdat daar de afstand tussen bestaande kennis en nieuwe kennis in termen van het aantal tussenliggende stappen het grootst is. Als zodanig zou de student, welis-waar onder begeleiding, zelfstandig of in teamverband in staat moeten zijn om op basis van verworven kennis en vaardigheden, nieuwe kennis te verwerven, te integreren in de al aanwezige kennis en dat eventueel om te zetten in een product binnen het domein communicatietechnologie. Daar waar de af-tand tussen bestaande en nieuwe kennis kleiner is, is een constructivistische aanpak wellicht niet zinvol.

Een derde argument voor een aanpak of cursusvorm als deze, is het gebruik van de vakgroep als instrument bij reguliere opleidingen. Een constructivistische aanpak representeert wellicht (de geromantiseerde) werk- en studieomgeving binnen een vakgroep of kenniskring. Meer concreet: een nieuwe ontwikkeling doet zich voor, ook voor docenten/onderzoekers, en de afspraak wordt gemaakt om die nieuwe ontwikkeling gezamenlijk aan te pakken. Gezamenlijk omdat dan gebruik gemaakt kan worden van reeds beschikbare ervaringen van de teamleden en voldoende (tijds)capaciteit bijeen wordt gebracht om een zinvolle invulling van de taken te realiseren (in dit geval ca. 200 uur per project) (Baas 2002)

Meer concreet kunnen zeven algemene doelstellingen aan constructivistische omge-vingen worden verbonden:

1. Ervaring opdoen met de constructie van kennis (de student is verant-woordelijk voor het leren).
2. Ervaring opdoen en waardering voor verschillende visies (standpunten en probleemoplossend).
3. Het studeren inpassen in een rea-listische en relevante context (authentieke leertaken).
4. Het bevorderen het 'eigendom van' en 'stem in' het leerproces (de student centraal en de docent als consultant)

5. Het inpassen van het leerproces in een sociale omgeving (bevorder samenwerking)
6. Het gebruik van verschillende presentatiemethoden bevorderen (ken-nis uiten en delen).
7. Bevordering van bewust kennis contructie proces (reflectie op handelen en denken)

Deze doelstellingen leiden tot het ontstaan van zelfsturende teams die zichzelf bewust sturen ten aanzien van hetgeen ze leren en hoe wordt geleerd. Wanneer constructivisme wordt toegepast in de context van groepsleren, instructie en Internet, wordt er in de regel gesproken van computer onder-teund, samenwerkend leren (Computer Supported Collaborative Learning, CSCL).

2. Raamwerk voor het ontwerp van CSCL

Vanuit de theorie bestaan er zes richtinggevende ontwerpvragen voor een CSCL-omgeving (Strijbos, Kirschner, Martens 2004).

1. Welke leerdoelen worden nagestreefd?
2. Op welke manier vindt de interactie plaats?
3. Welke soorten taken worden gedeinieerd?
4. Hoeveel sturing is noodzakelijk gelet op leerdoelen, verwachte interactie en taaktype?
5. Wat is de optimale groepsgrootte gelet op leerdoelen, verwachte interactie, taaktype en vóór-structurering?
6. Op welke manier kunnen computers het beoogde leren en de verwachte interactie het beste ondersteunen?

Elk van deze ontwerpvragen kan weer worden uitgesplitst naar deelvragen. Belangrijk is echter dat deze ontwerpvragen tot doel hebben om een cursusontwerper te stimuleren om dieper na te denken over een cursusontwerp; ze zijn geen checklist voor gegarandeerd succes!

2.1 De constructie van de cursus 'Topics in Communicatietechnologie'

In de loop van de jaren 2000 tot en met eind 2002 werd de cursus 'Topics in Communicatietechnologie' geconstrueerd (Baas 2003). De studielast van deze cursus is 100 studiebelastingsuren en is gepositioneerd in de master technical informatics. Het belangrijkste leerdoel is om de studenten te leren omgaan met een onvolledig, slecht geconstrueerd probleem (ill-structured) en nieuwe kennis te verwerven om dit probleem (ten dele) op te lossen. Naast de eis vanuit de constructivistische principes is ook gekozen voor groepswerk om een substantieel aantal uren in de oplossing van een probleem te kunnen steken. Met uitzondering van het groepswerk en de hoeveelheid studiebelastingsuren wordt de, in de cursus gecreëerde situatie als representatief gezien voor een belangrijk deel van de procesgang bij het afstuderen. Het betreft vooral de onderdelen: probleemafbakening, informatie vergaren en rapporteren. De cursusconstructie heeft nogal lang geduurd doordat het beoogde cursusmodel (CSCL) nogal afwijkt van de toen gangbare cursusmodellen (individueel, eigen tempo, schriftelijk, causaal en deterministisch). Bovendien ontbraken nog onderzoeken als (Strijbos, Kirschner, Martens 2004) en (Strijbos 2004) om de geformuleerde opvattingen over het ontwerp van de cursus wetenschappelijk te onderbouwen. In het begin moest vooral uitgegaan worden van intuïtie opgebouwd in jarenlange ontwikkeling en begeleiding van afstandsonderwijs. Dit laatste betekent dat het cursusontwerp niet helemaal de opvattingen

in de nu beschikbare wetenschappelijke bronnen volgt.

De nagestreefde leerdoelen
De nagestreefde leerdoelen zijn:
- Structuur kunnen aanbrengen in een vrij chaotische omgeving (ill-structured environment) door middel van een expliciete beschrijving van het te volgen proces, reflectie op bestaande kennis en vaardigheden, identificatie van nieuwe kennis en vaardigheden.
- Een topic kunnen inpassen in een mogelijke toepassing.
- Een brede en evenwichtige schriftelijke presentatie kunnen geven over de bestudeerde topics ten behoeve van medestudenten.
- Gemaakte keuzes, oplossingen of ontwerpen kunnen verdedigen, zowel beargumenteerd vanuit de inhoud als vanuit het gevolgde proces.
- De bestudeerde topics kunnen plaatsen binnen de context van huidige en toekomstige ontwikke-lingen.
- Initiatief tonen voor het aanpakken van onbekende problemen.
- Kennis opdoen m.b.t. het academisch proces in relatie met het afstuderen.

Samenvattend komen de leerdoelen neer op een decompositie van een probleem in deelproblemen, een keuze hieruit maken en de oplossing baseren op referenties in de literatuur.

De taakconstructie
In de cursus worden twee taken voorzien waarmee studenten kunnen reflecteren op hun ervaringen bij de eerste taak de resultaten van deze reflectie kunnen toepassen bij de uitvoering van de tweede taak. Op die manier wordt de student bewust gemaakt van zijn of haar leerervaringen. De taken zelf zijn van het 'ill-structured' type wat zoveel wil zeggen dat de uitkomst niet van te voren vaststaat en van de studenten een actieve houding eist. De taken zijn actueel en authentiek voor het vakgebied waardoor de intrinsieke motivatie van de studenten wordt bevorderd. De omvang en formulering van de taak is zodanig dat de studententeams alleen al op grond van beschikbare tijd gedwongen worden om een beargumenteerde keuze te maken welk deel van de taak wordt aangepakt en welk deel de studenten laten liggen. Op deze manier wordt ook de groepsdynamiek bevorderd. Studenten worden gestimuleerd om gebruik te maken van de publicaties van voorgaande studenten-teams. De teams worden gevraagd om een taak af te sluiten met een artikel op wetenschappelijk niveau. Voor het uitvoeren van de taken wordt gebruik gemaakt van litteratuuronderzoek. De aangeboden bronnen zijn onder andere die van de ACM en IEEE waarbij de artikelen zowel als middel, maar ook als voorbeeld dienen. Afhankelijk van het onderhandelingsresultaat met de docent kan in sommige gevallen de tweede taak een deeltaak (een verdieping) uit de eerste taak zijn.

De interactie
De aard van de interactie hangt af van de taken of rollen die de afzonderlijke teamleden uitvoeren. Met opzet zijn de aangeboden rollen zo gekozen dat er geen hiërarchische verhoudingen ontstaan waarbij teamleden zich kunnen verschuilen achter eerder verworven vaardigheden (bijvoorbeeld de rol van 'projectleider'). In tegendeel, de rollen zijn zodanig geconstrueerd dat er een positieve afhankelijkheid ontstaat (Strijbos 2004). Dat wil zeggen dat de groepsleden ieder voor zich verantwoordelijk worden gehouden voor het groepsresultaat en dat de groep verantwoordelijk is voor het leren van elk van zijn leden. De aangeboden rollen zijn:

- ‘communicator’: verantwoordelijk voor de communicatie met de docent
- ‘datacollector’: verantwoordelijk voor de (vóór)-selectie van de literatuur en het onderhouden van de kennis-bestanden
- ‘reporter’: verantwoordelijk voor het opstellen en de redactie van het artikel
- ‘projectplanner:’ verantwoordelijk voor de projectplanning en de voort-gang.

De rolbeschrijvingen zijn ontleend aan (Strijbos 2004) en zijn als bijlage bij dit artikel opgenomen. De aangeboden rolbeschrijvingen hebben ook tot doel dat een team eerder inhoudelijk aan de slag gaat in plaats tijd te besteden aan de verdeling en definiëring van taken.

De mate van sturing
Gezien de plek die de cursus in het curriculum inneemt, wordt een groot deel van het werk, en dus ook het leerproces naar het team gedelegeerd. Met andere woorden, het team wordt zelfsturend verondersteld, temeer daar de taak ook geen structurerende elementen bevat. Zelfsturing houdt ook in dat het eindresultaat in principe onvoorspelbaar is (het team is verantwoordelijk en niet de docent). Dit gegeven is in strijd met de gangbare ‘controle’-cultuur rondom cursussen en creëert een spanning tussen het vooraf regisseren van het cursusverloop of het team los te laten om op avontuur te gaan.
Bij deze cursus is voor een beperkte sturing gekozen. De cursus begint met een face-to-face startbijeenkomst waarin de opzet en doelstellingen van de cursus worden uitgelegd, de teamleden alvast met elkaar kennis kunnen maken en wordt de veronderstelde ‘flow’ in de cursus toegelicht. Bij de eerste opdracht worden de door het team voorgestelde aanpak en nadere precisering van het onderzoek, de tussenrapportages en het eindresultaat uitgebreid door de docent becommentarieerd. Bij de tweede opdracht is dit ‘toezicht’ aanzienlijk minder.
Een aantal ondersteunende documenten over de wijzen van literatuuronderzoek, het schrijven van een wetenschappelijk artikel, de voordrachten van de startbijeenkomst en de resultaten van andere teams staan in elektronische vorm op de gebruikte elek-ronische leeromgeving (ELO) als naslag. Met uitzondering van een voorbeeld van een zelfevaluatieformulier is verder afgezien van het verschaffen van sjablonen voor de tussenrapportages, procesverslagen en verdere structurering van het proces.

Optimale teamgrootte
Conventionele wijsheid en de beschikbaar-heid van rollen hebben geleid tot een stan-daardteamgrootte van 4 studenten. Het is nog niet voorgekomen dat twee rollen in één persoon werden gecombineerd zodat een teamgrootte van 3 ontstaat of een team van 5 personen waarin één persoon geen rol heeft. Uit oogpunt van planning zou deze mogelijkheid wel onderzocht moeten worden.

De elektronische leeromgeving
Bij de constructie van de cursus is niet uitputtend gezocht naar de meest geschikte ELO. De voorhanden zijnde ELO (Black-board) is gebruikt vanwege de (asynchrone) communicatiemogelijkheden email, discussiegroep en chat. Daarnaast heeft deze ELO allerlei instrumenten voor de procesafstemming zoals kalenders, agenda’s. Ook zijn er methoden voor bestand-uitwisseling.

3. Discussie en conclusie

Bij de start van de cursus in 2003 werden zes studenten en de docent/begeleider in een zogenaamde veldtoets (Stalmeier 2003) systematisch bevraagd naar hun ervaringen

met de cursus. Ook komen er gegevens beschikbaar uit de procesrapportages en het contact met de huidige studenten tot nu toe. Met deze laatste groep is de populatiegrootte 33 personen. De studenten zijn volwassenen die naast hun werk in deeltijd studeren.

De studenten geven unaniem een positief oordeel over het houden van een startbijeenkomst. Met name het face-to-face kennismaken met toekomstige teamleden en bij die gelegenheid alvast wat werkafspraken maken, wordt erg op prijs gesteld. De conclusie moet dan ook zijn in e-learning omgeving het onderdeel 'startbijeenkomst' een plek heeft wanneer studenten met elkaar in groepen moeten samenwerken. Toch moet er verder naar een elektronische vervanger van een startbijeenkomst met fysieke aanwezigheid worden gezocht omdat buitenlandse studenten (in dit geval drie) vanwege de afstand niet aan deze bijeenkomst konden deelnemen. Het blijkt dat deze studenten in vergelijking met andere studenten een moeizamer start doormaken.

Om aan het gebruik van Blackboard te wennen, wordt bij iedere sessie een elektronisch voorstellingsrondje georganiseerd waarin studenten foto's van zichzelf kunnen plaatsen en een aantal zaken over zichzelf kunnen publiceren. Geen van de studenten heeft uit zichzelf een verband gelegd tussen deze oefening en de face-to-face startbijeenkomst. Blijkbaar wordt de elektronische voorstellingsronde niet als adequate vervanger van deze startbijeenkomst gezien.

Alle studententeams volgen de voorgestelde rolverdeling, ondanks dat er een eigen, afwijkende rolverdeling is toegestaan. De belasting door de verschillende voorgestelde rollen blijkt verschillend te zijn, maar niet echt op bezwaren te stuiten. Er waren geen tekenen dat er in de teams 'meelifters' waren. In dat opzicht werkt één van de doelstellingen van de voorgestelde rolverdeling, creatie van positieve afhankelijkheid, goed.

Het ontbreken van een hiërarchie leidt volgens sommige studenten tot een besluitvorming die tijdrovend is en nogal eens voorbij gaat aan de wetenschappelijke feitelijkheid (compromis). Er werden overeenkomsten gezien met het 'poldermodel'. Deze studenten stellen een aan de praktijk ontleende rolverdeling, zoals bijvoorbeeld projectleider, meer op prijs. Het blijkt overigens dat degene die de rol van 'communicator' vervult, dikwijls ook de natuurlijke leider in het team wordt of is. Dus enige hiërarchie ontstaat van nature; vermoedelijk doordat langs deze communicatielijn de autoriteit van de docent doorstraalt. De conclusie is dat er enige hiërarchie ontstaat en wellicht een meer conventionele rolverdeling wenselijk zou kunnen zijn. Hoe dan de risico's van 'meeliften', 'doen waar je al goed in bent' en ondersteuning van constructivistische principes geminimaliseerd worden is nog onderwerp van verder onderzoek. Wel blijkt uit de literatuur (Strijbos 2004) dat er geen verschil in leereffecten bestaat tussen teams die volgens deze aangeboden rollen werkten en teams waarbij geen rolverdeling werd aangeboden (en dus waarschijnlijk met conventionele rollen gingen werken).

Ten opzichte van klassiek geconstrueerde cursussen is CSCL op basis van constructivistische principes geen causaal of deterministisch proces. Het proces is probalistisch, allerlei cursusonderdelen kunnen worden klaargezet maar het is een kwestie van afwachten en alert reageren of het proces loopt zoals het moet lopen. Temeer doordat een aantal zaken voor de docent niet zichtbaar zijn (communicatie) en er een spanning bestaat tussen het laten 'zelfsturen' van de teams en van buitenaf ingrijpen door de docent. Met andere woorden, voor de

docent ontstaat een managementprobleem. Tijdige en adequate communicatie en afstemming tussen team en docent blijkt essentieel. Dit proces laat in de tot nu toe gedraaide run's te wensen over. Ondanks de voorgestelde procesgang met tussentijdse mijlpalen (de flow) volgen de teams min of meer 'stilzwijgend' hun eigen weg. Dus wél zelfsturend, maar het product (onderzoek en artikel) behoeft nogal eens een flinke verbetering. Er is dus meer dan de verwachte sturing nodig. Maar hoe nu de sturing zodanig te doseren om het constructivistische in de cursus niet te verstoren en niet te vervallen tot dirigisme, is nog een punt van verder onderzoek.

Opvallend is dat de aangereikte bronnen van de ACM en IEEE nauwelijks worden geraadpleegd. Men haalt de informatie uit de vrij beschikbare bronnen op het Internet en incidenteel vanuit de papieren media. Dit heeft vermoedelijk te maken met de bereikbaarheid van deze bronnen. Studenten kunnen vanuit huis elektronisch de samenvattingen zien, maar vanwege de licentierechten niet de bijbehorende artikelen ophalen. Ophalen kan alleen via een geautoriseerde toegang op de studiecentra van de Open Universiteit Nederland. Creatie van een dergelijke toegang vanuit een thuis- of mobielstation is essentieel voor het functioneren van een cursus als Topics.

De gebruikte ELO (Blackboard) is wat betreft de bediening door docenten eenvoudig. Toelaten van studenten en indelen in groepen alsmede het plaatsen van extra ondersteunende documenten 'on the fly' met de bijbehorende aankondigingen verloopt snel en simpel. Ook studenten vinden snel hun weg in deze ELO. Een alom gevoeld nadeel is het ontbreken van realtime communicatie in de beschikbare implementatie van Blackboard. Recent zijn de studenten begonnen met het gebruik van gereedschappen voor spraakcommunicatie over Internet (Skype). De ervaringen hiermee zijn zodanig wisselend dat hierover nog geen conclusies kunnen worden getrokken. Ook is er nog geen ervaring opgedaan met de aangeboden 'building blocks' met een soortgelijke functie van Blackboard. Wel ontstaan door de communicatiebeperkingen van Blackboard nieuwe werkwijzen voor de studententeams. 'Groeidocumenten', short en long lists met gevonden litteratuur en het commentaar over de bruikbaarheid daarvan, uitwijken naar andere communicatiemiddelen (vooral MSN) en telefonische conferenties zijn voorbeelden hiervan. Wel wordt het ontbreken van een versiebeheersysteem bij BB als nadeel gezien.

De beoordeling van het eindresultaat is een teambeoordeling en in termen van onvoldoende of voldoende. Een verdere nuancering in de beoordeling ontbreekt omdat de mogelijke basis hiervoor bij de huidige versie van de cursus onzichtbaar is. Tot op heden heeft geen van de studenten deze beoordeling als 'unfair' betiteld. Het is echter te verwachten dat op onderdelen van de cursus toch een individuele beoordeling wordt gewenst. Een mogelijke ingang hiervoor zou het procesverslag kunnen zijn. Dit verslag heeft echter primair de functie om leerervaringen vast te leggen en een koppeling aan een beoordeling zou deze functie kunnen verstoren. Een andere mogelijkheid is een beoordeling van de (gelogde) communicatie. Zinnen zouden dan van een typering voorzien kunnen worden, bijvoorbeeld inhoudelijke bijdrage, procesbijdrage, voorstel, besluit, enzovoorts, maar de uitvoering daarvan is zeer bewerkelijk (Strijbos 2004). Met andere woorden, de beoordeling van teamwerk in deze situatie is nog onderwerp van verdere studie.

Referenties

- Alexander, Johanna (1999), *Collaborative design, constructivist learning, information technology immersion, & electronic communities: a case study*, California State University, Bakersfield, ISSN 1064-4326
- Baas, Koos (2003), *CommTech presentatie TiCT,* interne prestentatie voor de domeingroep Communicatietechnologie
- Baas, N. P. J. M. (2002), *Cursusplan 'Topics in communicatietechnologie' (T25311),* Open Universiteit Nederland
- Plugge, L.(2003), *ICT-voorzieningen in 2003,* in 'De vruchten plukken, Trends en Visie, Wetenschappelijke Technische Raad SURF, ISBN 90-74256-24-4
- Stalmeier, M (2003), *Veldtoets-resultaten Topics*, interne communicatie OUNL
- Jan-Willem Strijbos, Paul A. Kirschner, Rob L. Martens, (eds) (2004), *What We Know About CSCL*, ISBN 1-4020-7779-3, Kluwer Academic Publishers
- Strijbos, Jan-Willem (2004), *The effect of roles on cumputer supported collaborative learning,* Proefschrift Open Universiteit Nederland, ISBN 90-9018487, J. W. Strijbos, Heerlen

Bijlage: Rolbeschrijvingen en taken bij de cursus Topics in Communicatietechnologie

Onderstaande rolbeschrijvingen zijn ontleend aan: Strijbos, Jan-Willem, *The effect of roles on cumputer supported collaborative learning* (pp. 68-69), Proefschrift Open Universiteit Nederland, ISBN 90-9018487, 2004 J. W. Strijbos, Heerlen

Projectplanner

Responsibility: project planning and monitoring project progress

Activities:

Make an inventory of the role that group members prefer to perform and the amount of effort (time) that each member can invest on a weekly basis. You will communicate the roles and time effort via the planning sheet to all group members and the tutor, before 15 October.

You are responsible for recording all agreed on date's, activities or tasks, deadline's and current status of activities or tasks (finished or not) on the projectplanning sheet.

On a regular basis you will monitor deadline's and current status of tasks.

You will stimulate active participation of all team members, i.e. regular request ofr update's regarding activities or deadline's.

If necessary, or caused by sudden changes (for instance increased demands from your work), you will make an inventory of possible changes regarding planning or deadlines, record responses by group members and distribute the updated planning sheet among group members.

You are required to formulate an agenda for content discussion ('which aspects need to be discusses', 'what aspect have priority'), u will make an inventory of points suggested by team members and distributed the final discussion agenda.

You will initiate (and stimulate) discussion of literature suggestions provided, pre-selected literature and additional information sources ('Which sources are relevant?', 'How can certain information be used regarding the assignment?').

You will initiate discussion of comments to drafts and/ or proposed changes to the (draft)report.

In case team members prefer to distribute information sources and/ or domain specific aspects of teh assignment, among group members, you are required (in collaboration with the datacollector) to make an inventory and planning of the distribution pf information sources. This distribution will be recorded on the project planning sheet.

Communicator

Responsibility: communication with tutor and progress reports

Activities:

You are responsible to make an inventory of questions and problems that team members experience during the assignment, and for communicating these to the tutor and his/ her answer to all group member.

Each message intended for the tutor is indicated by tying 'Message to the tutor: followed by the question or problem that needs to be addressed)' in the subject line of the message.

Your tutor will only answer messages that are send by the 'communicator', the tutor will not answer questions or problems asked by other team members.

You will communicate the inventory by the 'Projectplanner' (i.e. role and effort (time) that can be invested on the project planning sheet) to your tutor before 15 October.

Every two weeks you will prepare a short progress report (maximum 1 page) of the communication between group members, discussion of content, communication with tutor etc. Use the progress report sheet.

In case the total communication appears too voluminous you will restrict your report to the main point that have been discussed, decisions taken and/ or developments in group collaboration.

You will send your report to your tutor, as well as uploading the report in the file

exchange section (keeping communication between group members transparent). In week 42 on monday 15-10 you will send the first report. The next report is required in week 44 on monday 29-10, etc. (every two weeks).
You will construct an archive of the content discussion about literature, difference between perspective, differences between domains, and various theories that are introduced and discussed. This overview will be included in each progress report.
You are responsible for submitting your groups' final report to your tutor.

Reporter

Responsibilities: Drafting and editing of the article

Activities:

You will prepare a first draft of the report and distribute this among team members. They are required to respond to this draft within a timeline that you have specified (e.g. five days), with comments, questions, reformulations, additional information, formulation of the text etc.
You will revise each draft according to comments provided by team members. You will distribute the next version among team members with another request for comments and suggestions.
You are responsible to make an archive of the different versions of the report.

Datacollector

Responsibility: literature pre-selection en collecting additional literature and maintaining shared knowledge base

Activities:

You will make pre-selection of literature provided. In this pre-selection you will indicate which information sources are most relevant and on which aspects of the case sufficient or relevant information lacks. You will distribute this pre-selection among group member with a request for suggestions on additional literature.
Based on all comments and suggestions by team members on your pre-selection you will adapt the list 'required literature and/ or information sources', for instance with library of internet sources.
You are responsible for distributing additional information sources, and/ or dividing sources according to individual expertise together with the team member responsible for project planning.

NIOC bestuur

Mr. J.A. (Hans) Frederik – voorzitter

j.a.frederik@wxs.nl

Drs. T.M. (Trudy) Berends – vice voorzitter

t.m.berends@pl.hanze.nl

Dr. R. (Rein) Smedinga – secretaris

r.smedinga@cs.rug.nl

Ir. C.A. (Kees) van Loon – penningmeester

c.a.van.loon@hhs.nl

Drs. H. (Henk) van Leeuwen

h.vanleeuwen@saxion.nl

P. (Peter) Tolen

p.tolen@zonnet.nl

Ir. J.A. (Jaap) Zwaan

jzwaan@xs4all.nl

Marijke de Wijs -- secretaresse

Drs. J.L.J. (Jos) Tolboom -- voorzitter pc

Samenstelling Programmacommissies

Algemene programmaleiding
- Jos Tolboom - Rijksuniversiteit Groningen
- Victor Schmidt - Hanzehogeschool Groningen

Programmacommissie WO
- René Bakker - Open Universiteit
- Jan Herman Verpoorten - Universiteit Utrecht
- Harm Bakker - Rijksuniversiteit Groningen
- Nico van Diepen - Universiteit van Twente
- Erik Barendsen - Katholieke Universiteit Nijmegen
- Johan Jeuring - Universiteit Utrecht
- Ria van Ouwerkerk - Technische Universiteit Eindhoven

Programmacommissie HBO
- Victor Schmidt - Hanzehogeschool Groningen
- Willem Prakken - Saxionhogeschool Enschede
- Ton de Bruyn - Saxionhogeschool IJselland
- Wim Hendriksen - Hogeschool Arnhem en Nijmegen
- Miranda Valkenburg - Fontys Hogescholen
- Wim Smit - Hogeschool Leiden

Programmacommissie MBO
- Mark Vlasblom - ROC ASA
- Marjan Vernooy - CITOgroep
- Sef Presser - ROC Leiden
- Jan Smit - Bve Raad
- Joep Swagemakers - ECABO
- Marten Ris - ROC Leiden

Programmacommissie VO
- René Franquinet - Arentheem College, Arnhem, Vereniging i&i
- Frans Peeters - St. Willibrordcollege, Goes
- Bert Zwaneveld - Open Universiteit
- Natasa Grgurina - SG Sevenwolden, Heerenveen
- Daphne Kerstens - Arentheem College, Arnhem
- Marjo Bollen - RSG Broklede, Utrecht
- Marja ter Wal - CC Schaersvoorde, Aalten

Programmacommissie Bedrijfsopleiders
- Henk de Bruyne - Capgemini
- Oscar Helfferich - LOI
- Rienke Voorburg - EXIN
- Lucielle de Bakker - Sogeti

Sponsors NIOC2004

ESSENT Kabelcom B.V.
http://www.essentkabelcom.nl

Microsoft
http://www.msita.net

Philips
http://www.philips.com

Sun Microsystems
http://www.sun.com/

Organiserende instellingen NIOC2004

Rijksuniversiteit Groningen
http://www.rug.nl

Hanzehogeschool Groningen
http://www.hanze.nl

Samenwerkingspartners

Algemeen

Informatie
http://www.informatie.nl/

Tinfon
http://www.tinfon.nl/

Ngi, platform voor ICT-professionals
http://www.ngi.nl/

Edict
http://www.edict.nl/

CODI

NIO, Nationale Informatica Olympiade
http://olympiads.win.tue.nl/nio/

WO

Open Universiteit
http://www.ou.nl

TU Eindhoven
http://w3.win.tue.nl/nl/

Universiteit van Amsterdam
http://www.uva.nl/

Universiteit Groningen
http://www.rug.nl/informatica

Universiteit Twente
http://www.ewi.utwente.nl/

VU Amsterdam
http://www.few.vu.nl/

HBO

Hanzehogeschool
http://www.hanze.nl/default.htm

HBO-I platform
http://www.hbo-i.nl/asp/intro.asp

Kennismanagement VIC

Nederlandse Toegepaste Informatica Lectoren (NTIL)

MBO

Loket MBO-ICT
http://www.loketmboict.nl/site/

ROC-I partners
http://www.roc-i-partners.nl/

VO

Vereniging I&I
http://www.ieni.org/

Bedrijfsopleidingen

Capgemini Academy
http://academy.capgemini.nl/

LogicaCMG Academie
http://www.logicacmg.com/

EXIN
http://www.exin.nl/

Gegevens van de auteurs

Auteurs zijn op alfabet opgenomen
(tussen haakjes de paginanummers van de betreffende artikelen)

B _____

Baas (105)

Ing N. P. J. M (Koos) Baas is universitair docent communicatietechnologie bij de faculteit Informatica aan de Open Universiteit Nederland en onder andere cursusontwikkelaar en cursusteamleider van de cursus Topics in Communicatie Technologie (TiCT) die volgens de principes van CSCL is ontwikkeld

Open Universiteit Nederland
Postbus 2960
6401 DL Heerlen

Koos.Baas@ou.nl

Boltjes (64)

Dr. E.G. (Elise) Boltjes is werkzaam als docent op de Noordelijke Hogeschool Leeuwarden bij de afdeling Engineering en bij de lerarenopleiding exacte vakken. Tevens is zij trainer/adviseur voorbeeldgestuurd onderwijs bij het Educatief Centrum Noord en Oost (ECNO).

e.g.boltjes@nhl.nl

Bruidegom (14)

B. (Ben) Bruidegom is verbonden aan het AMSTEL-instituut van de faculteit natuurwetenschappen, wiskunde en informatica (FNWI) van de Universiteit van Amsterdam en lid van het docenten team van de bachelor opleiding informatica van de UvA.

Universiteit van Amsterdam
AMSTEL-instituut, FNWI
Kruislaan 404
1098 SM Amsterdam

B.Bruidegom@uva.nl
www.science.uva.nl/~benb

D

Dijk (62)

Betsy van Dijk, Universiteit van Twente

bvdijk@cs.utwente.nl

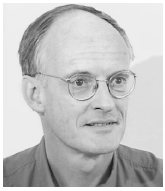
E

Everts (66)

Maarten Everts is student informatica aan de Rijksuniversiteit Groningen en lid van het informatica promoteam, zie pagina 66.

F

Feldbrugge (24)



Ir. F.H.J. (Frits) Feldbrugge is als docent informatica en als curriculumcoordinator verbonden aan de Informatica Communicatie Academie van de HAN.

Informatica Communicatie Academie (ICA)
Hogeschool van Arnhem en Nijmegen (HAN)
Ruitenberglaan 26
6826 CC Arnhem

Frits.Feldbrugge@han.nl

G

Groote (90)

Prof.dr.ir. J.F. (Jan Friso) Groote is hoogleraar voor Ontwerp en Analyse van Systemen en opleidingsdirecteur van de informaticaopleiding.

Technische Universiteit Eindhoven
Faculteit Wiskunde en Informatica
Postbus 513
5600 MB Eindhoven

j.f.groote@tue.nl

H

Heeren (73)

Drs. B.J. (Bastiaan) Heeren is assistent in opleiding aan de Universiteit Utrecht, bij de Software Technology groep van de faculteit Wiskunde en Informatica. Hij hoopt binnenkort zijn proefschrift te verdedigen.

Universiteit Utrecht
Faculteit Wiskunde en Informatica
Padualaan 14
3508 TB Utrecht

bastiaan@cs.uu.nl
www.cs.uu.nl/~bastiaan

J

Jansen (83)

Drs. A.G.J. (Anton) Jansen is als promovendus verbonden aan de Rijksuniversiteit Groningen, bij de Sofware Engineering en ARCHitecture (SEARCH) groep. Hij doet onderzoek naar ontwerpbeslissingen in software architecturen. Tevens is hij één van de twee oorspronkelijke ontwikkelaars van het Athena systeem.

Rijksuniversiteit Groningen
Instituut voor Wiskunde en Informatica
Postbus 800, 9700 AV Groningen

a.g.j.jansen@cs.rug.nl

Joukes (49, 55)

Drs. G.W.M. (Gertje) Joukes is senior projectmedewerker bij de VHTO (expertisebureau vrouwen in bèta/techniek), met als voornaamste aandachtsgebieden de aansluiting vo-mto en vo/mbo-bèta/technisch ho, en de instroom in bèta/technische opleidingen.

www.vhto.nl

K _____

Kaasenbrood (90)

E.J.S. (Eric) Kaasenbrood is als studentassistent aan het onderzoek naar denkniveaus bij algoritmen verbonden. Hij volgt aan de Technische Universiteit Eindhoven de Masteropleiding Computer Science Engineering.

Edenstraat 10B
5615 GA Eindhoven

e.j.s.kaasenbrood@student.tue.nl

Kockelkorn (30)

Ir. L. (Ludo) Kockelkorn is directeur van het Expertisecentrum ICT van de Faculteit ICT, Hogeschool Zuyd. Tot augustus 2004 was hij werkzaam als faculteitsdirecteur van HEAO ICT en gedurende de jaren 1999 t/m 2002 tevens projectleider van het landelijk project duaal ICT van de HBO Raad. .

L.Kockelkorn@hszuyd.nl

Koolen-Wijkstra (14)

Dhr. W.M. (Wouter) Koolen-Wijkstra is masterstudent *Logic and Computation* aan de Universiteit van Amsterdam en programmeur van SIM-PL i.s.m. het AMSTEL-instituut van de UvA, de faculteit natuurwetenschappen, wiskunde en informatica van de UvA en de Digitale Universiteit.

wmkoolen@science.uva.nl

Kuper (67)

Dr.ir. J. (Jan) Kuper is universitair docent aan de Universiteit Twente (afdeling Informatica) op de gebieden logica en functioneel programmeren.

UniversiteitTwente
Afdeling Informatica
Postbus 217
7500 AE Enschede

jankuper@cs.utwente.nl

L _____

Laman (66)

Timo Laman is student informatica aan de Rijksuniversiteit Groningen en lid van het informatica promoteam, zie pagina 66

Lei (49)

drs. A.H. (Hennie) van der Lei-Wichers was tot 2002 werkzaam bij de Hogeschool van Arnhem en Nijmegen als manager aansluiting vo/mbo-hbo. Zij houdt zich sinds 1986 bezig met het vergroten van de instroom van meisjes in technische hbo-opleidingen. Sinds 2002 is zij als adviseur verbonden aan ICA.

Leijen (73)

Dr. D.J.P. (Daan) Leijen is postdocteraal onderzoeker aan de Universiteit Utrecht, bij de Software Technology groep van de faculteit Wiskunde en Informatica. Hij doet onderzoek naar het ontwerp van functionele talen en typesystemen.

Universiteit Utrecht
Faculteit Wiskunde en Informatica
Padualaan 14
3508 TB Utrecht

daan@cs.uu.nl
www.cs.uu.nl/~daan

Loosdrecht (20)

J. (Jaap) van de Loosdrecht is coördinator van het Computer Vision Lab van de Noordelijke Hogeschool Leeuwarden en is daar tevens docent aan de opleiding Informatica.

Tesselschadestraat 12
8913 HB Leeuwarden,
tel. 058 – 2961 193

j.van.de.loosdrecht@tech.nhl.nl
www.engineering.tech.nhl.nl/computervision
www.engineering.tech.nhl.nl/engineering/personeel/loosdrec/vision_course/index.html

M

Mofers (96, 105)

Dr.ir. F. J. M. (Frans) Mofers is als universitair hoofddocent werkzaam bij de faculteit Informatica van de Open Universiteit Nederland. Hij is verantwoordelijk voor het domein Communicatietechnologie en actief in onderzoeksprojecten op het gebied van elektronische leeromgevingen

Open Universiteit Nederland
Postbus 2960
6401 DL Heerlen
tel: 045 576 24 21

Frans.Mofers@ou.nl

Muizelaar (9)

Drs. S.A. (Sander) Muizelaar is verbonden aan de Hogeschool van Utrecht (HvU) en is onderwijskundig adviseur bij Cetis, het expertisecentrum voor onderwijsinnovatie en ict van de HvU.

sander.muizelaar@hvu.nl

P

Passier (96)

Ir. H.J.M. (Harrie) Passier is als universitair docent werkzaam bij de faculteit Informatica van de Open Universiteit Nederland. Hij ontwikkelt cursussen op het gebied van communicatietechnologie en doet promotieonderzoek naar feedbackmechanismen in elektronische leeromgevingen.

Open Universiteit Nederland
Valkenburgerweg 177
6419AT Heerlen
tel: 045 576 2828

harrie.passier@ou.nl

Perrenet (90)

Dr.drs. J.C. (Jacob) Perrenet is aan de Technische Universiteit Eindhoven onderwijskundig medewerker bij de opleidingen Technische Informatica en Toegepaste Wiskunde. Verder werkt hij TU/e-breed aan Ontwerpgericht Onderwijs en Academische Vorming.

Technische Universiteit Eindhoven
Faculteit Wiskunde en Informatica
Postbus 513
5600 MB Eindhoven

j.c.perrenet@tue.nl

Plessius (9, 35)

Ir. H.A. (Henk) Plessius is verbonden is aan de Hogeschool van Utrecht (HvU) en programmamanager Picture, het alignmentprogramma van de ict-opleidingen van de HvU.

henk.plessius@hvu.nl

R _____

Ravesteyn (35)

Drs.ir. J.P.P. (Pascal) Ravesteyn MSc is verbonden aan de Hogeschool van Utrecht bij het Instituut voor Proces Innovatie, één van de kenniscentra van de Hogeschool.

Instituut voor Proces Innovatie, Hogeschool van Utrecht
Nijenoord 1, 3552 AS UTRECHT
Postbus 182, 3500 AD UTRECHT

pascal.ravesteijn@hvu.nl

S

Schoonman (40)

Dr. W. (Wouter) Schoonman is als lector werkzaam bij Saxion Hogescholen. Zijn leeropdracht is gericht op de ontwikkeling en toepassing van Assessment ten behoeve van competentiegericht onderwijs (zowel wat betreft studenten als docenten). Daarnaast heeft hij een eigen praktijk op de grens van ICT en HRM, genaamd PsyTech (www.psytech.nl).

Saxion Hogescholen
Postbus 70.000
7500KB Enschede
Tel: 053 - 4871650

w.schoonman@saxion.nl

Smits (55)

Eliane Smits van Waesberghe is als projectmedewerker verbonden aan het VHTO (Expertisebureau meisjes/vrouwen en bèta/techniek)

www.vhto.nl

Starre (66)

Laurens van der Starre is student informatica aan de Rijksuniversiteit Groningen en lid van het informatica promoteam, zie pagina 66.

T

Tönissen (46)

Drs. R.A.M. (René) Tönissen, MME is adjunct-directeur van het instituut voor Informatica van de Hogeschool van Amsterdam; vice-voorzitter HBO-I;
voorzitter werkgroep profiel bachelor of ICT.

Hogeschool van Amsterdam,
Weesperzijde 190,
1097 DZ Amsterdam,
tel.: 020-5951632

r.a.m.tonissen@hva.nl
(HBO-I-bureau: info@hbo-i.nl)

Trommer (66)

Sylvia Trommer is student informatica aan de Rijksuniversiteit Groningen en lid van het informatica promoteam, zie pagina 66.

V

Veen (57)

Drs. M.J.P. (Maarten) van Veen is universitair docent bij de faculteit Informatica en als projectleider van het project Informatievaardigheden in het onderwijs verbonden aan het Ruud de Moor Centrum van de Open Universiteit.
Zie voor meer informatie de weblog: http://www.ou.nl/open/mjv/
Ruud de Moor Centrum
Open Universiteit
Postbus 2960
6401DL Heerlen
tel. 045-5762373

Maarten.vanveen@ou.nl

Vodegel (9)

Drs. F.A. (Frans) Vodegel is verbonden aan de Hogeschool van Utrecht (HvU) en is programmamanager SPIoN, het sectorale programma ict van de Digitale Universiteit.

frans.vodegel@hvu.nl

W

Wieling (66)

Martijn Wieling is student informatica aan de Rijksuniversiteit Groningen en lid van het informatica promoteam, zie pagina 66.

Wopereis (57)

Drs. I.G.J.H. (Iwan) is als onderwijstechnoloog verbonden aan het Onderwijstechnologisch Expertisecentrum (OTEC) van de Open Universiteit Nederland. Zijn professionele interesse gaat uit naar Instructional Design, Information Problem Solving, E-learning en het gebruik van ICT binnen lerarenopleidingen.

OTEC
Open Universiteit Nederland
Postbus 2960,
6401DL Heerlen
tel. 045-5762825

Iwan.wopereis@ou.nl

www.ou.nl/info-alg-innovatieonderzoek/InstructionalDesign/Personen/iwan_wopereis.htm

Z _____

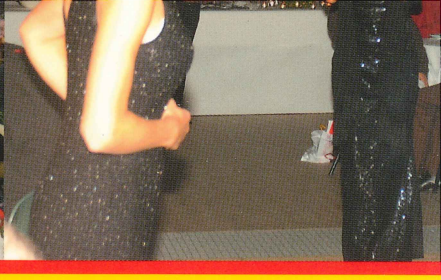
Zwaneveld (62)

Prof.dr. G. (Bert) Zwaneveld is als hoogleraar professionalisering van de leraar verbonden aan de Open Universiteit en het Ruud de Moorcentrum van de OU. Dit centrum houdt zich bezig met de professionalisering van de onderwijsgevenden.

Ruud de Moorcentrum van de Open Universiteit Nederland
Postbus 2960
6401 DL Heerlen
Tel. 045-5762294
Fax: 045-5762115

bert.zwaneveld@ou.nl





NIOC 2004 is de zevende in een rij van congressen over informaticaonderwijs.
Centraal in het congres staat het onderwijs in de informatica en informatie- en communicatietechnologie (ICT) in alle lagen van het onderwijs: wetenschappelijk, hoger beroeps, middelbaar beroeps en voortgezet onderwijs en bij de bedrijfsopleidingen. Naast deze zogenaamde ontmoetingsplaatsen zijn ruim vijfentwintig samenwerkingspartners bereid gevonden mee te denken en mee te werken aan de totstandkoming van NIOC 2004 en NIOC uit te laten groeien tot een instituut voor kennisuitwisseling, ook buiten de feitelijke congressen om.

NIOC 2004 kent een breed spectrum aan bijdragen uit elk van de hierboven genoemde ontmoetingsplaatsen. Hiervan is een groot aantal in deze proceedings terug te vinden.

UITGEVERIJ Passage

ISBN 90 5452 124 4