

# 1.1 Ontwerpen van activerende colleges

Koen Langendoen

## Samenvatting

Mijn competenties op het gebied van het ontwerpen van activerende colleges zijn gebaseerd op de ‘didactische training informatica’ (IT&C) die ik gevolgd heb in 2001 en het vervolgens geheel vernieuwen van het Compilerbouw vak (in4020<sup>1</sup>) in 2002. Toen was het nog een keuzevak uit het 4<sup>e</sup> jaar Technische Informatica (en verplicht voor 4<sup>e</sup>-jaars elektro studenten in de Computer Engineering variant), maar inmiddels is het een verplicht vak in de Masters Computer Science en de Masters Computer Engineering. De voertaal is nu dus Engels. De vernieuwing van het vak bestond uit het kiezen van een boek, ontwikkelen van ondersteunende powerpoint slides, en aanpassen van het bijbehorende practicum ontwikkeld op de VU. Om te beoordelen hoe het college door de Delftse studenten gewaardeerd werd, heb ik een enquête gehouden. De resultaten hebben geleid tot diverse (kleine) aanpassingen.

## 1 Doelstellingen

**Competentie:** De docent is in staat de doelstellingen van zijn cursus helder te formuleren, te plaatsen binnen het programma van vakgroep en faculteit, de latere beroepspraktijk en het studietraject van de student.

### 1.1 bewijsstukken

Het compilerbouw college maakt (verplicht) onderdeel uit van zowel de Masters Computer Science<sup>2</sup> als de Masters Computer Engineering<sup>3</sup> aangezien compilers een verbindende schakel vormen tussen software (computer programma’s, informatica) en hardware (elektrotechniek). Het vak staat geroosterd in het 4<sup>e</sup> jaar en geeft de student een kijkje “onder de motorkap” van de implementatie/executie van computerprogramma’s, hetgeen een dieper begrip oplevert van de mogelijkheden en beperkingen van allerlei taalconstructies die aan de basis liggen van veelgebruikte moderne programmeertalen (C++, Java, Prolog, Haskell).

Omdat de beschrijvingen van beide Masters opleidingen geen concrete leerdoelen voor het vak compilerbouw geven heb ik zelf de lijst in Figuur 1 opgesteld. Om deze leerdoelen te bereiken heb ik gekozen voor een formule met naast een hoor/werkcollege een groot practicum waarbij studenten met een complete reference compiler moeten werken. Dit practicum is essentieel om studenten de link tussen theorie en praktijk te laten leggen.

### 1.2 validatie

Appendix A beschrijft de diverse bronnen (enquêtes, cijfers, enz.) waarmee ik mijn competenties betreffende het ontwerpen, geven en toetsen van activerende colleges valideer.

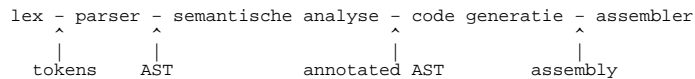
<sup>1</sup>[http://blackboard.icto.tudelft.nl/bin/common/course.pl?course\\_id=\\_6011\\_1&frame=top](http://blackboard.icto.tudelft.nl/bin/common/course.pl?course_id=_6011_1&frame=top)

<sup>2</sup>[http://academics.its.tudelft.nl/nl/info2002/Master\\_TI.pdf](http://academics.its.tudelft.nl/nl/info2002/Master_TI.pdf)

<sup>3</sup>[http://academics.its.tudelft.nl/nl/info2002/Master\\_Et.pdf](http://academics.its.tudelft.nl/nl/info2002/Master_Et.pdf)

Informatica studenten moeten na het succesvol afronden van het vak compilerbouw de volgende leerdoelen bereikt hebben:

1) Inzicht hebben in de structuur van een compiler:



en wat de interfaces zijn tussen de diverse onderdelen.

- 2) Begrijpen hoe een lexical analyser generator werkt.
  - weten dat LEX een token specificatie accepteert
  - weten dat LEX een FSA (Finite State Automaton) genereert
  - weten hoe de FSA geconstrueerd wordt (character moves, epsilon moves), en dit ook kunnen uitvoeren op een voorbeeld tokenspecificatie.
- 3) Begrijpen hoe een parser generator werkt.
  - weten dat er diverse klassen bestaan
    - \* top-down: recursive descent, LLGEN
    - \* bottom-up: YACC
  - en de verschillen kennen: LL vs. LR
  - weten dat LLGEN een LL(1) grammatica als input heeft, en een recursive descent parser als output genereert.
  - weten dat YACC een LALR(1) grammatica als input heeft
  - weten dat YACC een PDA (Push-Down Automaton) genereert, en deze ook kunnen construeren (mbv. shifts, reductions en epsilon-moves).
- 4) Het gereedschap kennen om semantische analyse te doen:
  - attribuut grammatica's, en bijbehorende evaluatie mechanismen
  - symbolic interpretation
  - dataflow equations
- 5) Enig begrip hebben van code generatie: het omzetten van een annotated AST in een lineaire instructiestroom (assembly).
  - code selection (ladder sequences)
  - register allocation (graph coloring)
  - instruction orderingweten dat code generatie een NP compleet probleem is en dat er heuristieken bestaan op diverse niveaus
  - AST nodes
  - basic blocks (dependency graphs)
  - procedure level
- 6) Begrijpen hoe de gegenereerde code zich runtime gedraagt
  - routine activation (stack)
  - scope resolution (dynamic vs static link)
- 7) Weten dat er naast een compiler ook een runtime support nodig is, vooral bij moderne talen die een hoog abstractie niveau hebben:
  - geheugen beheer (garbage collection)
  - object-oriented (dispatch tables)
  - functioneel (graph reducer)
  - logisch (backtracking)en begrijpen hoe deze onderdelen werken.
- 8) Praktische ervaring hebben met een (kleine) compiler en de standaard compilerbouwtools
  - lex
  - yacc

Figuur 1: Leerdoelen compilerbouw.

Uit de enquêtes blijkt dat slechts een enkele student moeite heeft met de pragmatische aanpak van het vak en liever meer theorie zou zien, het overgrote deel vindt het practicum weliswaar moeilijk maar ook uitdagend en leerzaam. Enkele quote's:

- “Het is interessant om te leren hoe het ‘onder de motorkap’ werkt”
- “Focus too much on practical work; do not hesitate to increase the theoretical level of the course.”
- “Weer es een praktisch hardcore vak tussen alle theoretische onzin”

### 1.3 zelfreflectie

Inmiddels hebben er twee lichten studenten (2002 + 2003) het vak compilerbouw gevolgd. Uit de tentamenresultaten is mij gebleken dat de theorie in het algemeen goed overkomt. Echter, veel studenten hebben moeilijkheden met het toepassen van deze kennis in de praktijk. Het practicum vraagt veel meer tijd dan de begrote studielast. (De oorzaken die daaraan ten grondslag liggen komen uitgebreid in sectie 2.1<sup>4</sup> aan bod.) Maar zowel de studenten als de collega docenten zijn ervan overtuigd dat het nuttig is om ook eens een groot, complex stuk software (bijv. een compiler) te doorgronden aangezien veel studenten daar later mee de bijbehorende problemen geconfronteerd zullen worden in hun technische loopbaan.

## 2 Voorkennis

**Competentie:** De docent is op de hoogte van de bij de studenten aanwezige voorkennis die van belang is voor het te doceren vakgebied/cursus.

### 2.1 bewijsstukken

Het compilerbouw vak (boek + practicum) is origineel op de VU ontwikkeld voor 5<sup>e</sup> jaars informatica studenten. Een korte inventarisatie leerde mij dat de noodzakelijke voorkennis (programmeren in C, programmeertalen, software engineering, formele methoden) reeds in de eerste 3 jaar behandeld wordt in het Delftse informatica programma.

Het compilerbouw vak is op één belangrijk punt aangepast tov. het origineel op de VU: het practicum is drastisch ingekort om de studielast van 6 studie punten (VU) terug te brengen naar 4 (Delft).

### 2.2 validatie

De enquête resultaten (Appendix A) laten zien dat de meeste studenten inderdaad de benodigde voorkennis bezitten, maar dat er toch diverse studenten deelnamen met matige voorkennis. Dit waren voornamelijk elektrotechniek studenten, waarvoor het vak in 2002 opeens verplicht was gesteld (hiervoor was het een keuze vak).

### 2.3 zelfreflectie

Toen in 2002 het practicum na enkele weken inleidend college begon werd het al snel duidelijk dat de meeste elektrotechniek studenten noodzakelijke praktische programmeervaardigheden misten. Dat lag niet aan de studenten zelf maar aan de aansluiting in het curriculum tussen de elektro Bachelors en de informatica Masters: de elektro studenten hebben te weinig geprogrammeerd en bovendien niet in de juiste taal in vergelijking met de informatica studenten. Ik heb verschillende opties overwogen om het gat in de voorkennis te dichten:

---

<sup>4</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/2.1/Geven-van-activerende-colleges.pdf>

1. practicumopdracht van compilerbouw aanpassen (minder programmeren).
2. stoomcursus programmeren in C inlassen in het compilerbouwcollege.
3. curriculum elektro aanpassen met extra programmeeronderwijs.

In eerste instantie heb ik ad-hoc besloten optie 1 te volgen om de lichte 2002 zonder al te veel problemen er doorheen te krijgen. Vervolgens heb ik met de collega's van elektro overlegd over een structurele oplossing, aangezien ik graag de formule van "handen uit de mouwen" zou handhaven. Men was het er mee eens dat het doen van een grote opdracht de student beter voorbereidt op de arbeidsmarkt dan het inkrimpen tot een kleine vingeroefening. Vervolgens is de oplossing gevonden in het invoeren een extra vak "systems programming in C" aan het begin van het 4<sup>e</sup> studie jaar van het elektro curriculum (optie 3). Dit was een goed te verdedigen aanpassing omdat het voor elektro studenten uiterst nuttig is om ervaring met de programmeertaal C te hebben aangezien die bij voorkeur gebruikt wordt om (elektronische) hardware te besturen. Inmiddels is duidelijk geworden dat ook de tweede lichte elektro studenten (met kennis van C) nog moeite heeft met het practicum: de studenten kunnen het wel, maar het kost ze veel te veel tijd. Daarop is besloten toch een aangepast practicum te maken: minder programmeerwerk en een onderwerp (code generatie) dat dichter tegen de hardware (processor) aan ligt.

In 2003, toen compilerbouw ook verplicht werd voor alle informatica studenten, bleek dat zij ook vaak praktische vaardigheden missen die nodig zijn om het practicum vlot af te ronden (zie enquête resultaten<sup>5</sup>). In mijn optiek vereist dit een verbetering van het programmeeronderwijs in de eerste twee jaar vd informatica opleiding. Dat veranderproces is in gang gezet. Om de gang van zaken op de korte termijn te stroomlijnen heb ik inmiddels het practicum enigszins verlicht door enkele randzaken (oa. Makefile's) kado te doen en inhoudelijke assistentie aan te bieden tijdens de practicum uren.

De hele exercitie met het herhaaldelijk aanpassen van het curriculum en practicumopdracht om zo in te spelen op veranderende voorkennis vd studenten heeft veel extra inspanning vd docent (en studenten!) gevraagd. Dit toont dus wel aan hoe belangrijk het is om nauwkeurig te weten wat studenten nu werkelijk weten en kunnen.

### 3 Werkvorm

**Competentie:** Kan juiste onderwijsvorm kiezen uit hoorcollege, werkcollege, practicum.

#### 3.1 bewijsstukken

Mijn onderwijsmotto is "al doende leert men", vandaar dat ik voor het Compilerbouw vak gekozen heb voor de combinatie hoorcollege en practicum. De resultaten voor het practicum bepalen voor 70% het eindcijfer, het afsluitend tentamen weegt voor 30% mee. Om de studenten kennis te laten maken met diverse facetten van het vak is er gekozen voor een practicum waarbij ze een complete 'reference compiler' krijgen voor een klassieke programmeertaal, die ze vervolgens moeten aanpassen om moderne taalconstructies te ondersteunen. Om de werklast aanvaardbaar te houden, werken studenten in groepjes van twee. Verder is er tijdens het practicum een student assistent aanwezig om te helpen met allerlei praktische problemen mbt de reference compiler en bijbehorende tools.

Het (interactieve) hoorcollege volgt in grote lijnen het "Modern Compiler Design"<sup>6</sup> boek, maar brengt diverse extra's:

**quiz vragen** over de stof vd vorige week, bedoeld om studenten "wakker te schudden", maar die ook als oefenmateriaal voor het tentamen dienen.

<sup>5</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/enquete-resultaten.pdf>

<sup>6</sup><http://www.cs.vu.nl/~dick/MCD.html>

**extra voorbeelden** die het materiaal in het boek complementeren.

**motiverende vragen** die de studenten bij de stof betrekken, bijv. “wat zijn de problemen met deze aanpak?”, “hoe kunnen we dit oplossen?”, “geef eens een voorbeeld uit de praktijk”.

**korte oefeningen** waarin meestal gevraagd wordt een algoritme toe te passen op een ander geval, en waarvan de antwoorden vervolgens plenair besproken worden.

**practicum instructie** ( $2 \times 45$  min.) waarin extra aandacht gegeven wordt aan de theorie die nodig is voor beide practicumopdrachten.

### 3.2 validatie

De enquête resultaten (Appendix A) geven aan dat de informatica studenten over het algemeen enthousiast zijn over het vak: nuttig *en* leuk (door het practicum). De elektro studenten vinden het vak vooral moeilijk door hun gebrek aan voorkennis; het elektro curriculum is inmiddels aangepast voor 2003 en er wordt nu een andere practicumopgave (code generatie) ontwikkeld die beter aansluit bij de (hardware) interesses vd elektro studenten. Het interactieve hoorcollege werd goed beoordeeld: juist tempo, juiste moeilijkheidsgraad oefeningen, en aangename hoeveelheid interactie. Enkele quote's:

- “nodigt uit tot het volgen van colleges (uniek voor mij :-)”
- “Is the hardest course I ever followed; lectures are the most fun part”
- “It is very hard for EE students”

### 3.3 zelfreflectie

Het volgen van de basis cursus didactiek voor informatica docenten (docent Toine Andernach) heeft een positieve invloed gehad op de vorm van het hoorcollege. Ipv. het klassieke hoorcollege geef ik nu een soort werkcollege met veel vragen (en antwoorden van studenten!) en meestal twee oefeningen waarin de stof die net behandeld is nu zelf door de studenten moet worden toegepast (“Hebben jullie dit begrepen? Mooi, dan volgt hier een oefening”).

Door het oefenen blijkt dat het vaak toch niet zo simpel was als het leek ... Daarom loop ik tijdens het maken vd oefening rond om aanwijzingen te geven; dat lukt alleen bij studenten die naast de trap of vooraan zitten want dan kan ik *ongevraagd* op hun papier kijken. Na circa tien minuten vraag ik om antwoorden. Om de drempel te verlagen mag de student dicteren, terwijl ik zijn antwoord op het bord uitwerk. Als een oplossing fout is wordt dat meestal al door andere studenten opgemerkt zonder mijn tussenkomst. Zo niet, dan wijs ik expliciet op het pijnpunt en vraag om een verbetering. Zo komen we gezamenlijk tot een goed antwoord.

Gezien de antwoorden van de studenten denk ik dat het practicum zijn doel goed bereikt: het werkt motiverend om iets praktisch met de verworven kennis te doen. Het tentamen gebruik ik om er voor te zorgen dat de kennis die niet bij het practicum aan bod komt, toch nog eens extra bestudeerd wordt.

## 4 Studiemateriaal

<b>Competentie:</b> Kan studiemateriaal/syllabus maken, boek kiezen.
--

## 4.1 bewijsstukken

Het studiemateriaal voor het Compilerbouw college bestaat uit een boek<sup>7</sup>, waarvan ik een co-auteur ben, een stel bijbehorende powerpoint slides<sup>8</sup>, die gebruikt worden tijdens het college, en een practicumopdracht<sup>9</sup>. Dit alles is ook te vinden op de uitgebreide blackboard site<sup>10</sup> die ik voor het vak ingericht heb.

Wat betreft de keuze van het boek merk ik nog op dat 1) het klassieke standaard werk (het ‘Drakenboek’ van Aho, Sethi en Ullman) uit 1986 toch echt verouderd is omdat het geen moderne programmeertaalconstructies behandelt, en 2) dat het recente werk van de grootste “concurrent” Appel (Modern Compiler Implementation, 1998) uitblinkt in oppervlakkigheid. Het is niet voor niets dat collega D. Grune vd VU na diverse boeken gebruikt te hebben besloten had om dan maar zelf een boek te schrijven.

## 4.2 validatie

De enquête resultaten (Appendix A) laten zien dat studenten tevreden zijn over het boek en de college handouts. Ze hadden wel het één en ander aan te merken op de practicum handleiding (“te beknopt”); deze is inmiddels aangepast en uitgebreid.

Het gebruikte boek<sup>11</sup> doet het ook internationaal gezien goed: het staat meestal genoteerd in de top-5<sup>12</sup> van best verkopende compilerbouw boeken. Het verouderde Drakenboek staat helaas nog altijd aan kop.

## 4.3 zelfreflectie

Het ontwikkelen van de college slides heeft ontzettend veel tijd gekost. Ik schat dat ik zo’n 3 à 4 dagen per college bezig geweest ben met het mechanisch vervaardigen vd slides. Ik had natuurlijk minder fraaie slides kunnen maken, maar ik heb toch de moeite genomen omdat:

1. duidelijke en geanimeerde slides beter overkomen bij de studenten tijdens het college.
2. de slides nu als handout meegegeven kunnen worden, en ze de studenten van nut zijn bij het voorbereiden op het tentamen.
3. het verhaal dat ik als docent moet houden nu zodanig gedetailleerd ondersteund wordt dat het opnieuw geven weinig voorbereiding vergt.
4. deze complete set slides nu beschikbaar gesteld kan worden (via de uitgever) aan docenten wereldwijd die overwegen over te stappen naar ons MCD boek en niet de tijd hebben om zelf slides te maken.
5. ik een perfectionist ben.

Omdat ik consequent na afloop van elk college de slides bijgewerkt heb (fouten verbeterd, extra commentaar toegevoegd, kleine modificaties aan de oefeningen doorgevoerd, enz.) heb ik in het 2<sup>e</sup> jaar gemerkt dat het derde voordeel inderdaad opgeld doet: met één keer de slides en notities bestuderen voor aanvang vh college zit alle stof weer in mijn hoofd. Of mijn gelikte slides inderdaad de college’s aantrekkelijker maken is lastig te beoordelen. Feit is dat de studenten tevreden zijn over mijn colleges, en ook de handouts goed beoordeeld worden.

<sup>7</sup><http://www.cs.vu.nl/~dick/MCD.html>

<sup>8</sup><http://www.pds.twi.tudelft.nl/~koen/compilerbouw/2003/rooster.html>

<sup>9</sup><http://www.pds.twi.tudelft.nl/~koen/compilerbouw/2003/practicum.html>

<sup>10</sup>[http://blackboard.icto.tudelft.nl/bin/common/course.pl?course\\_id=\\_6011\\_1&frame=top](http://blackboard.icto.tudelft.nl/bin/common/course.pl?course_id=_6011_1&frame=top)

<sup>11</sup><http://www.cs.vu.nl/~dick/MCD.html>

<sup>12</sup><http://www.bestbookdeal.com/cgi-bin/browse.cgi?topic=4422>

## 5 Studiewijzer

**Competentie:** Kan studiewijzer samenstellen.

### 5.1 bewijsstukken

De uitgebreide blackboard site<sup>13</sup> fungeert als studiewijzer. Studenten kunnen er alle benodigde informatie (college handouts, practicumopdracht, reference compiler, oefententamen, enz.) vinden die nodig is om het vak te doen. De gang van zaken bespreek ik kort tijdens het eerste college<sup>14</sup>.

### 5.2 validatie

De enquête resultaten (Appendix A) laten zien dat de studenten tevreden zijn met de informatie op blackboard.

### 5.3 zelfreflectie

Uit de tevredenheid vd studenten maak ik op dat een aparte studiewijzer niet nodig is als de informatie on-line beschikbaar is. De tijd die gemoeid is met het schrijven van een studiewijzer kan dus beter aan iets anders besteed worden, bijv. het vervaardigen van duidelijke college slides.

## 6 Lesplan

**Competentie:** Kan een lesplan maken.

### 6.1 bewijsstukken

Bij het opzetten van het lesmateriaal voor compilerbouw heb ik eerst een ruwe indeling gemaakt van wanneer welk stuk stof behandeld zou moeten worden. Dit is vastgelegd in het college rooster<sup>15</sup>. (Om te zorgen dat ik de stof die nodig is voor het doen van de beide practicum opdrachten op tijd behandel, hanteer ik een iets andere volgorde dan het boek.) Vervolgens heb ik per week een grof lesplan gemaakt dat het stramien volgt van 2 × 45 minuten college, met in elk deel een oefening (ca. 10 à 15 min.) waarin de studenten gevraagd wordt de zojuist behandelde stof actief toe te passen (meestal op een iets ander voorbeeld). Elk college start met een korte herhaling vd stof vd week ervoor voorzien van een quiz vraag (tentamen niveau) om de studenten weer “in te swappen” en effectief voort te kunnen bouwen op die stof. Dit opstarten vraagt ca. 5 min. Tenslotte heb ik voor elke week het generieke lesplan omgezet in een concrete powerpoint presentatie, die voorzien is van diverse ondersteunende aantekeningen. Voor het geven vh college lees ik deze ‘notes’ uitgebreid door en stap door de presentatie om niet verstrikt te raken in m’n eigen animatie’s. (Niets vervelender dan iets uit te leggen en er bij de volgende muisklik achter te komen dat je nog een prachtige ondersteunende animatie gemaakt had).

Een mooi voorbeeld van zo’n geïntegreerd lesplan is dat van week 4<sup>16</sup>. Deze les was vrij gecompliceerd omdat er op het bord met de hand een uitwerking moest plaats vinden die de powerpoint presentatie ondersteunt. Let ook op alle aantekeningen over de vragen

<sup>13</sup>[http://blackboard.icto.tudelft.nl/bin/common/course.pl?course\\_id=\\_6011\\_1&frame=top](http://blackboard.icto.tudelft.nl/bin/common/course.pl?course_id=_6011_1&frame=top)

<sup>14</sup><http://www.pds.twi.tudelft.nl/~koen/compilerbouw/2003/week1.pdf>

<sup>15</sup><http://www.pds.twi.tudelft.nl/~koen/compilerbouw/2003/rooster.html>

<sup>16</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/notes4.pdf>

(en de bijbehorende antwoorden) die ik aan de studenten stel om ze te laten nadenken over het hoe en waarom.

## **6.2 validatie**

Uit de enquêtes (Appendix A) blijkt dat de meeste studenten erg positief staan tov. het interactieve college, en dus (indirect) het lesplan. De positieve houding vd studenten tov. college kan ook afgeleid worden uit de opkomstcijfers die ik bijgehouden heb (Appendix A) en waaruit blijkt dat veel studenten tot en met het laatst de colleges (actief) volgen.

## **6.3 zelfreflectie**

In mijn geval van  $2 \times 45$  min. college kom ik uit op een netto spreektijd van ca.  $2 \times 30$  min; de rest van de tijd ( $2 \times 15$  min.) gaat op aan oefeningen maken en vragen beantwoorden. Het gevolg is dat ik iets minder uitgebreid op de stof kan ingaan, maar dat studenten wel ervaren dat de zaken moeilijker zijn dan ze lijken (een algoritme simpelweg herhalen valt niet mee!).

Bij het bedenken vd oefeningen die ik tijdens het college laat maken, heb ik geprobeerd een inschatting te maken vd moeilijkheid uitgedrukt in oplostijd (variërend van 3 tot 8 minuten). De praktijk heeft uitgewezen dat elke oefening ongeveer een kwartier in beslag neemt. In geval van een makkelijke oefening heeft iedereen 'm af. In het geval van een moeilijke oefening, die meestal uit twee delen bestaat, begin ik vaak als de snelste studenten klaar zijn. Dat betekent dat de anderen nog niet alles afgerond hebben, maar wel ruim de tijd gehad hebben om erover na te denken. Sommige moeilijke oefeningen heb ik na de 1<sup>e</sup> keer aangepast, maar anderen heb ik gehandhaafd: ook de goede studenten moeten af en toe uitgedaagd worden. Het alternatief van nog meer tijd voor de moeilijke oefeningen uit te trekken (verandering lesplan) heb ik snel verworpen want dan zou ik of moeten bezuinigen op het lesmateriaal of op de interactie met de studenten, en beide onderdelen wil ik graag handhaven; ca. 33% vd tijd gaat nu op aan oefeningen, en dat vind ik ruim voldoende.

## A Validatie bronnen

De kwaliteitszorg van het onderwijs is niet de sterkste kant van de opleiding Technische Informatica. Er worden voorzichtige initiatieven ontplooid omdat te verbeteren. Zo worden de vakken uit de Bachelorsfase standaard geëvalueerd middels een SENSOR enquête onder de studenten, hetgeen enige informatie oplevert maar alleen vanuit het gezichtspunt vd student, en niet van collega's (experts). De Masters vakken, waaronder mijn compilerbouw college, zijn dit jaar (2002-2003) voor het eerst geëvalueerd middels interviews met studenten ten overstaan van de betrokken docenten. Helaas kon ik daarbij niet aanwezig zijn en bleek er slechts 1 student te zijn komen op dagen.

Om toch inzicht te verkrijgen in mijn functioneren als (beginnend) docent heb ik een enquête gehouden tijdens één vd laatste colleges in 2002, toen ik het Compilerbouw vak voor de eerste keer doceerde. Er hebben 18 studenten een enquêteformulier ingevuld (zie de resultaten<sup>17</sup>). Omdat ik de feedback zinvol vond (en diverse aanpassingen heb doorgevoerd) heb ik in 2003 weer een enquête afgenomen. Ditmaal 32 reacties (zie de vragen<sup>18</sup> en antwoorden<sup>19</sup>). Verder spreek ik tijdens de pauzes geregeld met studenten die vragen hebben over het college en practicum, of die alleen maar klagen over de hoeveelheid praktisch werk. Ook ben ik op het in 2003 ingevoerde practicum olv. student assistenten wezen kijken en luisteren naar de studenten die net de cijfers van opgave 1 terug gekregen hadden.

Om de aantrekkelijkheid van mijn colleges te peilen heb ik de opkomstcijfers genoteerd. In 2002<sup>20</sup> is het me gelukt om de kleine groep studenten grotendeels vast te houden. In 2003<sup>21</sup>, toen compilerbouw een verplicht vak werd, was het verloop groter.

De laatste bron van informatie betreft de resultaten (practicum + tentamen) van de studenten: eindcijfers 2002<sup>22</sup> en 2003<sup>23</sup>. In 2002 heb ikzelf al het praktische werk nagekeken. In 2003 is dat gebeurd door student assistenten. De (matige) kwaliteit vd oplossingen is een aardige indicatie van hoe moeilijk de studenten het practicum vinden. De resultaten van het mondeling tentamen (2002) waren over het algemeen goed, die van het schriftelijk (juist/onjuist) tentamen<sup>24</sup> (2003) een stuk minder.

---

<sup>17</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/enquete-resultaten.pdf>

<sup>18</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/questionnaire\2003.pdf>

<sup>19</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/questionnaire-results.txt>

txt

<sup>20</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/opkomst02.txt>

<sup>21</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/opkomst03.txt>

<sup>22</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/eindcijfers-02.txt>

<sup>23</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/eindcijfers-03-vj.xls>

<sup>24</sup><http://rama.pds.twi.tudelft.nl/~koen/bko/in4020/2003-06-27-cijfers.pdf>