E.W. Dijkstra (1930-2002)

Jan Kraak
j.kraak@rc.rug.nl

Gestructureerd en tegelijk correct programmeren

Op 6 augustus jl. overleed in Nuenen op 72-jarige leeftijd prof. dr. Edsger W. Dijkstra, ongetwijfeld Nederlands belangrijkste informaticus van de 20e eeuw. Hij leed al geruime tijd aan kanker.

Dijkstra's ideeën over gestructureerd programmeren vonden gretig ingang bij sommigen binnen deze universiteit. Zijn methode voor correct programmeren wordt op veel plaatsen gedoceerd. In dit artikel zal ik, vanuit praktisch perspectief, wat vertellen over het werk van Dijkstra en enkele herinneringen ophalen.

20-80 regel

De 20-80 regel kent iedereen, zij het misschien niet bij naam. Bijvoorbeeld 20% van alle mensen drinkt 80% van alle geproduceerde bier. En omgekeerd drinkt 80% van alle mensen slechts 20% van alle geproduceerde bier. De verhouding kan ook 10-90 zijn, het gaat om het globale idee. De regel geldt voor veel menselijke consumptiepatronen, iedere winkelier is er mee vertrouwd.
Toegepast op software verklaart de regel waarom een programma, ondanks veel programmeerfouten, toch tevreden gebruikers kan hebben. De meeste gebruikers passen namelijk slechts een klein, goed uitgetest, deel van alle mogelijkheden toe, waardoor ze niet in contact komen met de fouten in het ongebruikte deel. Zo zijn de programma's van de firma Microsoft zeker niet foutloos, maar toch is Bill Gates er de rijkste mens van de wereld mee geworden.
De meeste mensen hebben zich neergelegd bij het feit dat programma's nu eenmaal fouten hebben. Net zo aanvaarden makers en gebruikers van Perzische tapijten een enkele verkeerde knoop. Ik las eens dat er soms met opzet een foute knoop in een tapijt wordt aangebracht omdat 'alleen God perfect is'.

De uitdaging van correct programmeren

Met dit soort redeneringen moet je niet aankomen bij de door software-fouten getergde 20% van de gebruikers, bij gedreven programmeurs en al helemaal niet bij informatici voor wie het een kwestie van beroepstrots is om zo foutloos mogelijk te programmeren. Ze zullen wijzen op programma's die absoluut correct moeten zijn, zoals programma's die een kernreactor beveiligen of navigatiesystemen van vliegtuigen.
Zij weten maar al te goed dat het schrijven van foutloze programma's erg lastig is, gezien de enorme complexiteit ervan: programma's van enkele miljoenen regels code zijn tegenwoordig geen uitzondering. Maar ze kennen ook goede gereedschappen en methoden om de uitdaging van correct programmeren aan te gaan. Het is vooral Dijkstra geweest die veel heeft bijgedragen tot de theorievorming en tot methoden om programma's zo foutloos mogelijk te maken.

Semaforen

In een artikel over een dag met Nederlandse computerpioniers in Delft (zie Pictogram april 2000), heb ik Dijkstra reeds geïntroduceerd als de 'de eerste programmeur' van Nederland. Tijdens zijn studie theoretische natuurkunde ging hij in 1952 op het Mathematisch Centrum in Amsterdam werken. Met J. A. Zonneveld bouwde hij de allereerste compiler voor de goed ontworpen programmeertaal ALGOL 60. In 1962 werd Dijkstra hoogleraar wiskunde aan de TU Eindhoven waar hij onder meer het THE Multiprogramming System ontwierp voor de X8 computer. Door toepassing van de seinpaal-metafoor vond hij een elegante oplossing voor de samenwerking van meerdere processen, zoals gebruikersprogramma's en randapparaten, bijvoorbeeld printers. Iedere informaticastudent kent tegenwoordig semaforen. Van 1973 tot 1984 was hij nog maar één dag in de week verbonden aan de TU. De rest van de week was hij Research Fellow van Burroughs en kon hij, verlost van allerlei sores, volledig zijn eigen gang kon gaan.

'Notes on Structured Programming'

In Dijkstra's Eindhovense tijd gingen veel wetenschappers met computers werken. Ze programmeerden er zorgeloos op los, meestal in de ook toen al veel gebruikte programmeertaal FORTRAN. Voor administratieve toepassingen werd COBOL gebruikt. Hij zal waarschijnlijk wel eens programma's van deze nieuwe categorie gebruikers onder ogen hebben gekregen en hij zal er waarschijnlijk van hebben gegruwd. Te meer omdat hij aan zichzelf de hoogste eisen stelde en diep doordrongen was geraakt van de moeilijkheid van programmeren.

Op grond van deze ervaringen schreef hij in 1969 zijn zeer opmerkelijke 'Notes on Structured Programming'. Niet alleen de inhoud was opvallend, maar vooral de stijl. Terwijl iedereen toen nog leerde dat je het woord 'ik' zo veel mogelijk moest vermijden, vooral in wetenschappelijke publicaties, gebruikte hij het woord bijna onbeschaamd vaak. Hier sprak iemand vanuit zijn diepste overtuiging. Er werd de indruk gewekt dat hier grote wijsheden werden verkondigd, die je niet kon negeren. In EWD-1308 (zie verderop) staat dat Dijkstra de 'Notes' na een depressie schreef, als een vorm van therapie.

Voorbeeld van een FORTRAN-fragment met 'logische spaghetti'

```
c  IF (X) 1,2,3 betekent:
c  spring naar label 1 als x < 0,
c  spring naar label 2 als x = 0
c  spring naar label 3 als x > 0

      IF (ISS)10,10,20
   20 IF (IX-IXO)12,21,12
   21 IF (IY-IYO)12,22,12
   22 IF (IS-ISO)12,23,12
   23 CALL PLALC(AM,NMAX,IRC)
      GO TO 73
   10 IF (IX2)13,50,13
   13 IF (IX2-MT)19,19,50
   19 IF (IY2)11,50,11
   11 IF (IY2-NT)12,12,50
   12 IF(TIND(IX2,IY2,1))     GO TO 73
      IF (CV-AM(IX2,IY2))206,206,5
  206 IF (IDX**2+IDY**2-1)213,6,213
  213 DCP=(AM(IX,IY)+AM(IX2,IY)+
     + AM(IX,IY2)+AM(IX2,IY2))/4.0
      IF (DCP-CV)5,217,217
  217 IF (INX(IS-1))214,215,214
  214 IX=IX+IDX

C .......
```

'The GOTO statement considered harmful'

Op het gevaar af zijn redeneringen te trivialiseren, zal ik trachten een paar ideeën uit dit werk onder woorden te brengen:

- Programmeren is zeer lastig, het vergt een grote concentratie. Daarvan moeten we ons bewust zijn, en daarom moeten we ons bescheiden opstellen en goed gebruik maken van onze beperkte mentale eigenschappen en naar eenvoud streven. We moeten denkwijzen gebruiken waarin we goed zijn, zoals logisch redeneren en abstractie.
- Door overvloedig gebruik van sprongen in een programma is het dynamische gedrag moeilijk uit de tekst te halen. Dijkstra gebruikte de term 'logische spaghetti' voor dit soort programma's. Het programmeerhulpmiddel om een sprong te maken, het GOTO-statement dat prominent in FORTRAN aanwezig is, moet daarom in de ban worden gedaan.

Over dit aparte probleem publiceerde Dijkstra in 1968 een korte notitie met de wat potsierlijke titel 'The GOTO statement considered harmful'. Velen kennen van Dijkstra alleen deze titel. Vooral in Japan heeft de titel bevreemding gewekt, omdat GOTO daar een bekende achternaam is. Op een conferentie stond eens een kleine Japanner op. Hij boog beleefd naar Dijkstra en vroeg hem: 'Why am I considered harmful?'. U raadt het al: dat was professor GOTO. In een andere versie van deze anekdote schrijft prof. GOTO het artikel 'Dijkstra considered harmful'.

- Het gebruik van GOTO-statements is te vermijden door IF-THEN-ELSE, FOR-WHILE etc. instructies te gebruiken. Aldus ontstaan er *gestructureerde* programma's. Thans heeft elke taal structureringsinstructies.
- Van ons abstractievermogen moeten we gebruik maken bij het top-down ontwerpen van programma's.

Academische vorming vereist

In het midden van de jaren 70 bezocht ik met enkele collega's, waaronder Adri den Arend, Johan Kelders, Doeko Homan en Doeke de Vries, die thans werkzaam is bij Wiskunde, een studiedag over gestructureerd programmeren met Dijkstra als één van de sprekers. Voor een muisstille zaal - met vrijwel allemaal mannen - vertelde hij met zachte stem een poëtisch verhaal over een fladderende vlinder.

Wat hij daarmee probeerde te verduidelijken, is me echter ontgaan. Duidelijker was zijn sneer naar COBOL: hij verwachtte dat de verzekeringsmaatschappijen problemen zouden krijgen door hun in COBOL geschreven programma's, zodat ze COBOL zouden afzweren. Hij zei ook dat een programmeur een academische opleiding moest hebben. Toen een serieuze programmeur, maar geen academicus, hier tegen bezwaar maakte, nuanceerde hij deze uitspraak.

Een andere spreker was G. A. Kroes, werkzaam bij Shell, die net SHELTRAN had gemaakt voor gestructureerd programmeren. Door een preprocessor werden de SHELTRAN-commando's in het snel werkende FORTRAN omgezet. Kroes was een grote bewonderaar van Dijkstra en had aan zijn muur de spreuk 'In den beginne was Dijkstra' hangen, zo vertelde hij. SHELTRAN werd in gebruik genomen bij Sterrenkunde, waar het nog steeds gebruik wordt in het vermaarde programma GIPSY (Groningen Image Processing System). Bij Sterrenkunde-medewerker Hans Terlouw hangt een portret van Dijkstra op het prikbord. Soms kijkt hij er naar, in de hoop een goedkeurend knikje van de meester te krijgen.

Herprogrammering KOMPLOT

In 1975 was ik intensief bezig met het opnieuw programmeren van het grafiekenprogramma KOMPLOT dat toentertijd binnen de hele universiteit en ook daarbuiten veel werd gebruikt om grafieken te maken. Daarom vielen de ideeën over gestructureerd programmeren in me 'als God's woord in een ouderling'. KOMPLOT was in ALGOL 60 geprogrammeerd, maar dat werd niet verder ondersteund op de toen nieuwe CDC-computer. Daarom kreeg ik de unieke kans om het bestaande programma geheel opnieuw op te zetten.

Als programmeertaal leek het toen ook al veel gebruikte FORTRAN een goede keuze, maar daar

kleefden de door Dijkstra genoemde bezwaren aan. SHELTRAN lag voor de hand, maar daar heb ik toch maar van afgezien omdat dat weer een extra hulpmiddel, met alle problemen van dien, introduceerde. Toen heb ik een eigen manier van gestructureerd programmeren in FORTRAN bedacht, waarbij ik met de hand IF-THEN (-ELSE)-constructies en andere structureringsinstructies vertaalde in GOTO's of met commentaar simuleerde. Ook paste ik top-down design toe: ik begon met een klein, goed werkend, programma en vulde gaandeweg de ontbrekende functionaliteit aan. Zo ontstond, in een tijd dat FORTRAN nog geen goede structureringsinstructies had (versie FORTRAN 66), toch een redelijk gestructureerd programma. Tot op de dag van vandaag heeft KOMPLOT vrijwel ongewijzigd gefunctioneerd.

'A discipline of programming'

Terwijl veel programmeurs nog gefascineerd waren door gestructureerd programmeren, was Dijkstra en zijn groep in Eindhoven al weer lang bezig met een nieuw aspect: het bewijzen dat een programma(fragment) ook inderdaad doet wat je ervan verwacht. Dat is niet simpel, want testen is niet voldoende. Een veel aangehaalde uitspraak van Dijkstra is dat je met testen alleen kunt aantonen dat een programma fouten heeft, maar niet dat het *zonder* fouten is. Over correct programmeren publiceerde hij in 1976 'A discipline of programming'. Het is lastig om in een paar zinnen de kern van dit boek uit te leggen, maar ik doe toch een poging.

Om te kunnen spreken van de correctheid van een programmafragment 'S', moet worden gespecificeerd onder welke preconditie 'Q' de uitvoering van 'S' eindigt in de gewenste postconditie 'P'. Hierbij zijn 'P' en 'Q' logische uitdrukkingen in termen van de variabelen van 'S'.

Het correctheidsbewijs van de programmaspecificatie, genoteerd als {Q} S {P}, is in het algemeen lastig, vooral als 'S' is samengesteld uit meerdere statements. Daarom voerde Dijkstra de zwakste beginvoorwaarde in, de zgn. *weakest precondition* wp(S,P), met bijbehorende rekenregels, waardoor correctheidsbewijzen beter uitvoerbaar worden. Een voorbeeld: de functie **sqrt** om een wortel te trekken in het statement Y:=**sqrt**(X); werkt alleen voor positieve argumenten, de preconditie is X >=0. De postconditie is X=Y*Y.

Het boek geeft veel voorbeelden van deze methode. Het is veel lastiger te lezen dan de 'Notes', met name vanwege de afwisseling van ogenschijnlijke trivialiteiten en diepzinnigheden.

Correct programmeren in Groningen

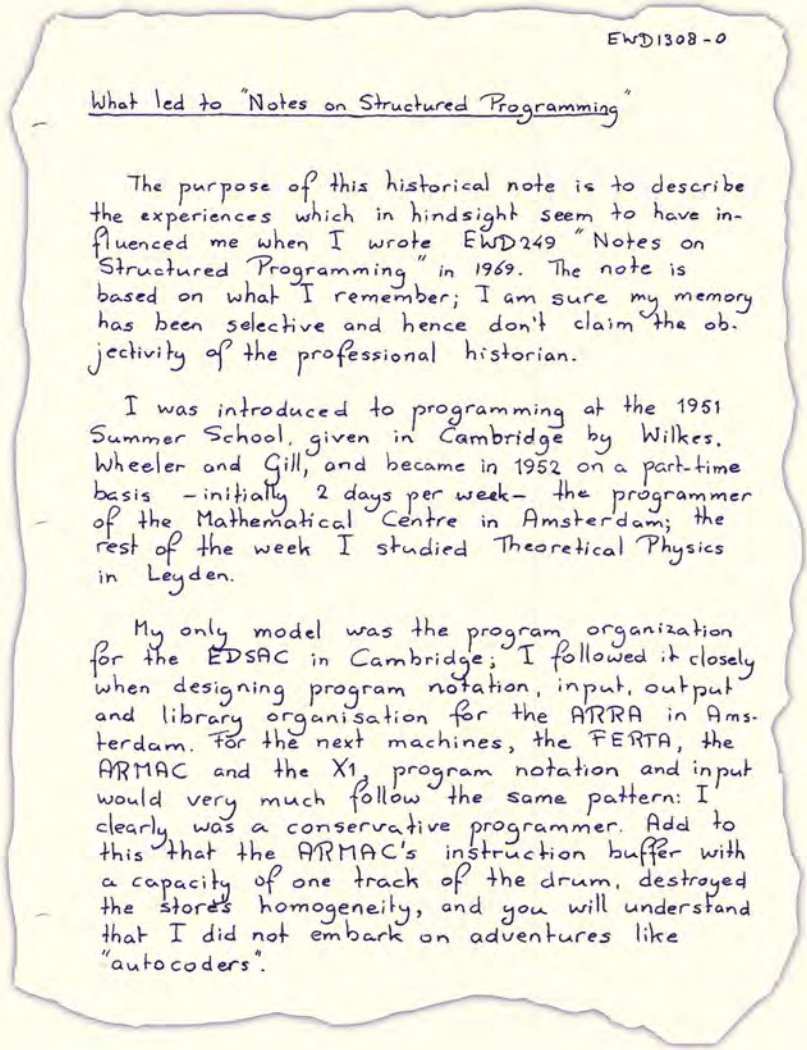
In de academische wereld heeft Dijkstra veel aanhang. In Groningen doceerde zijn leerling prof. dr. J. van de Snepscheut correct programmeren. Thans worden deze colleges onder meer gegeven door prof. dr. W. H. Hesselink. In zijn inaugurale rede 'Specificatie van Berekening' uit 1995 staat dat het doorzettingsvermogen vereist om correct te leren programmeren. Als student moet je namelijk beginnen met eenvoudige voorbeelden, waarvoor strenge correctheidsbewijzen niet nodig zijn. Maar je moet je daar toch doorheen worstelen, om ze later zelf te kunnen uitvoeren voor ingewikkelde programma's.

Hesselink illustreert hoe lastig de uitvoering van een correctheidsbewijs is voor een 'echt' probleem. Op zichzelf waren de uit te voeren bewijsstappen nog niet zo moeilijk, maar door de enorme hoeveelheid stappen dreigde hij de draad kwijt te raken. Hij paste toen met succes de stellingbewijzer NQTHM toe, die de administratie voor hem deed.

Omdat ze zoveel tijd kosten, maar ook omdat ze lang niet triviaal zijn, hebben grondige correctheidsbewijzen tot nu toe weinig ingang gevonden in de praktijk. Vaak beperkt men zich tot kleine, kritische programmafragmenten. Doeke de Vries, die de ontwikkelingen op programmeergebied altijd goed bijhoudt, liet me zien hoe hij, via informele pre- en de postconditie's van C-functies, zorg draagt voor hun correctheid. Waarschijnlijk zijn er meer manieren om Dijkstra's ideeën in de praktijk toe te passen.

Als Socrates

Naast zijn wetenschappelijke publicaties, die veel succes hadden in de academische wereld, werd Dijkstra bij een groter publiek ook



EWD1308-0

What led to "Notes on Structured Programming"

The purpose of this historical note is to describe the experiences which in hindsight seem to have influenced me when I wrote EWD249 "Notes on Structured Programming" in 1969. The note is based on what I remember; I am sure my memory has been selective and hence don't claim the objectivity of the professional historian.

I was introduced to programming at the 1951 Summer School, given in Cambridge by Wilkes, Wheeler and Gill, and became in 1952 on a part-time basis – initially 2 days per week – the programmer of the Mathematical Centre in Amsterdam; the rest of the week I studied Theoretical Physics in Leyden.

My only model was the program organization for the EDSAC in Cambridge; I followed it closely when designing program notation, input, output and library organisation for the ARRA in Amsterdam. For the next machines, the FERTA, the ARMAC and the X1, program notation and input would very much follow the same pattern: I clearly was a conservative programmer. Add to this that the ARMAC's instruction buffer with a capacity of one track of the drum, destroyed the stores homogeneity, and you will understand that I did not embark on adventures like "autocoders".

bekend om de prikkelende uitspraken die hij van tijd tot tijd deed over de praktijk van het programmeren en ander computergebruik. Voorbeelden daarvan staan bijvoorbeeld in 'How do we tell truths that might hurt?' uit 1975:

- 'Programming is one of the most difficult branches of applied mathematics; the poorer mathematicians had better remain pure mathematicians'.
- 'The easiest machine applications are technical/scientific computations'.
- 'FORTRAN – "the infantile disorder" –, by now nearly 20 years, is hopelessly inadequate for whatever computer application you have in mind today: it is now too clumsy, too risky, and too expensive to use'.
- 'The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence'.
- Etc.

Dijkstra zal, als ex-gymnasiast, bij het doen van dit soort uitspraken waarschijnlijk de rol van de Griekse filosoof Socrates voor ogen hebben gestaan die zijn omgeving ook tartte. Maar Dijkstra is niet, zoals Socrates ongeveer 2400 jaar eerder, veroordeeld tot het drinken van de gifbeker.

EWD-notes

Dijkstra was zeer productief. Veel van zijn ideeën heeft hij privé gepubliceerd in zijn serie EWD-notes die als een soort kettingbrief onder een grote schare bewonderaars werden verspreid. Een deel is met vulpen geschreven in een keurig handschrift. Als grap heeft iemand dat handschrift eens via TeX door een computer laten genereren.

In de, in totaal 1317, EWD-notes staan niet alleen wetenschappelijke resultaten, maar ook reisverslagen, colleges, allerlei opinies, een terechtwijzing van een nieuw lid van zijn dinsdagmiddag-club, een briefje aan een afstudeerder, etc. Hij schrijft ook ergens dat de 'considered harmful'-titel het werk is van Niklaus Wirth.

Door hun stijl, voor wie daar van houdt, zijn de EWD-notes vaak zeer leesbaar. Sinds enige tijd zijn ze via internet te lezen, dankzij de inspanningen van de universiteit van Texas in Austin, waaraan Dijkstra van 1984 tot 1999 was verbonden.

Links:

www.rug.nl/hpc/people/pioniers
www.cs.utexas.edu/users/EWD