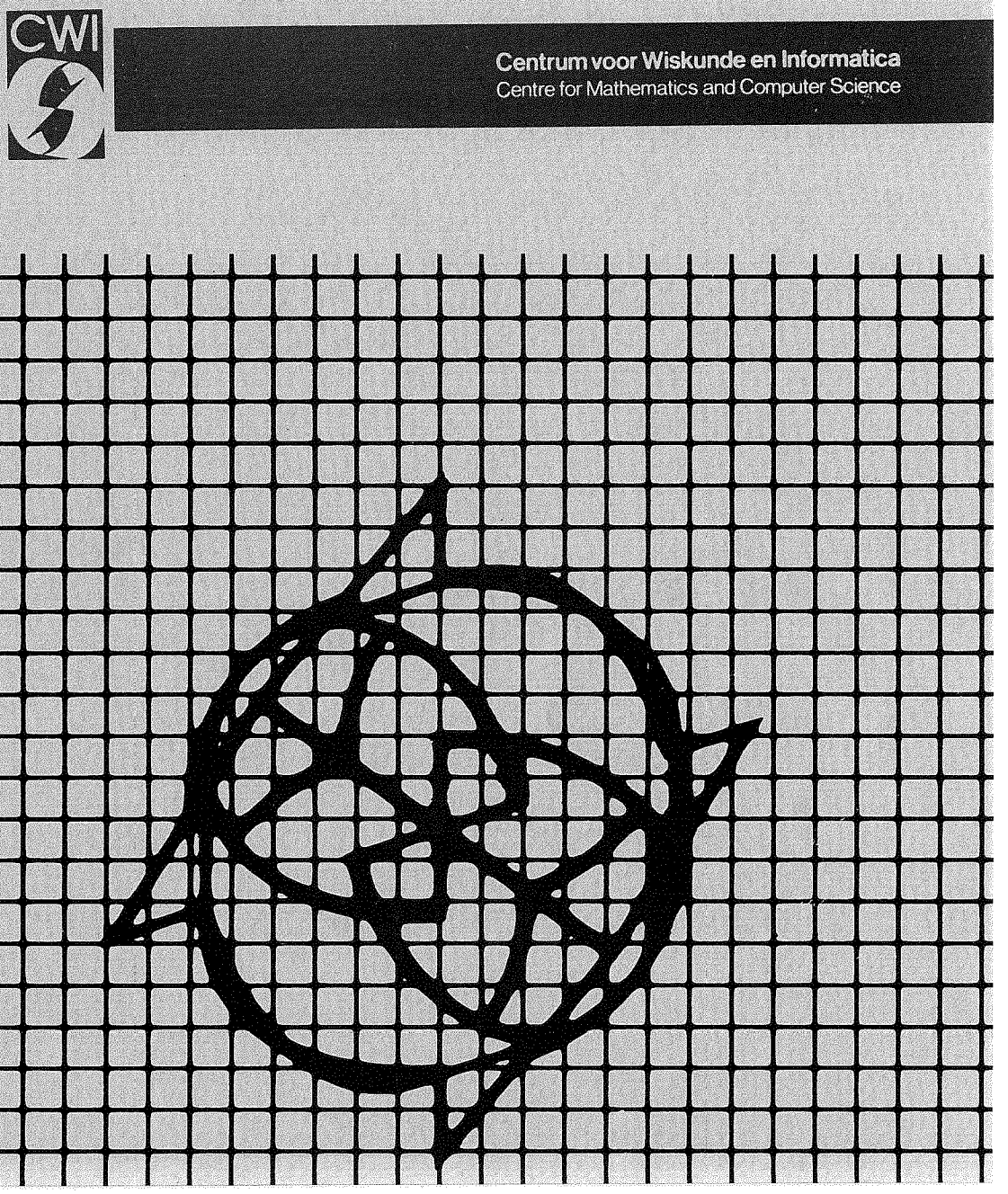
Programmeertechnologie



Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

Programmeertechnologie

Een wijdverbreide opvatting is, dat voor het succesvol schrijven van programma's kennis van een of meer programmeertalen voldoende is. Anders gezegd: de problemen, die men bij het programmeren ontmoet, zouden door de ontwerpers van programmeertalen al voorzien en opgelost zijn.

Deze opvatting is grotendeels onjuist. De taak van de programmeur – het overbruggen van de afstand tussen probleem en programmeertaal – vereist inzicht in beide. Maar terwijl de taal meestal nog wel onder de knie te krijgen is, geldt dat lang niet altijd voor het op te lossen probleem, eenvoudig omdat het te moeilijk of in zijn omvang overstelpend is. Het maken van een bedrijfs- of informatiesysteem, bijvoorbeeld, is zó moeilijk, dat niemand op dit moment kan zeggen dat hij over alle vereiste inzichten beschikt. De Informatica rekent het dan ook tot haar taak het inzicht in deze – en andere – problemen te vergroten. Overigens weerhoudt gebrekkig inzicht slechts weinigen ervan zich aan grote programmeerprojecten te wagen met als gevolg dat de uiteindelijke producten vaak onnodig ingewikkeld en kwalitatief beneden de maat zijn.

Die onderdelen van de Informatica die tot de *Programmeertechnologie* (of *Programmeerwetenschap*) gerekend kunnen worden pogen verbetering in deze situatie te brengen, zonder zich daarbij aan een specifiek probleemgebied te binden. De centrale gedachte daarbij is de programmeeractiviteit onderdeel te maken van een zoveel mogelijk gesystematiseerd ontwerp- en implementatieproces. Dit heeft geleid tot een nieuwe, veel geadverteerde, misvatting. Het zou voor het succesvol schrijven van grote programma's voldoende zijn om – naast kennis van een programmeertaal – te beschikken over een ontwerpmethodologie. Maar het feit dat volledige systematisering van het ontwerp- en implementatieproces in het algemeen uitgesloten is en dat het specifieke probleemgebied vooralsnog buiten beschouwing blijft, betekent dat de oplossingen die vanuit de Programmeertechnologie worden voorgesteld slechts betrekkelijke waarde hebben. Het zijn niet de wondermiddelen waarvoor ze moeten doorgaan. Niettemin kan het methodisch ontwerp van software – mits gecombineerd met inzichten in de specifieke problematiek geleverd door andere takken van informatica en uiterst intensief denkwerk van de programmeur(s) – in vele gevallen leiden tot een noodzakelijke en beslissende kwaliteitsverbetering van het afgeleverde product.

De belangrijkste ontwikkeling op dit moment en een die zich nog geruime tijd zal voortzetten is de ontwikkeling van *specificatietalen*. De bedoeling is in een zo vroeg mogelijk stadium van het ontwerpproces de mogelijkheid te hebben "abstracte" programma's te schrijven, waarvan de correctheid zo mogelijk evident is en die in ieder geval bepaalde formeel

bewijsbare eigenschappen hebben, maar die misschien nog volstrekt niet executeerbaar zijn. Deze abstracte programma's worden dan in achtereenvolgende stappen al of niet met assistentie van de computer "geconcretiseerd" tot een executeerbaar programma. De bedoeling is alleen concretiseringen toe te laten, die correctheidbehoudend zijn, zodat het eindresultaat gegarandeerd correct is relatief ten opzichte van de beginspecificatie. Volledig automatische concretisering is in het algemeen uitgesloten, al streeft de Programmeertechnologie juist op dit punt voortdurend naar terreinwinst. In hoeverre hier succes te behalen valt zal de toekomst moeten leren. Hoe dit ook zij, hier ligt een evident aansluitingspunt met het gebied van de Kunstmatige Intelligentie.

De ontwikkeling van specificatietalen is de natuurlijke voortzetting van de in 1954 met FORTRAN begonnen lijn van de hogere programmeertalen. De concretisering van abstracte specificaties is te beschouwen als generalisatie van het compilatieproces, met dien verstande, dat menselijk ingrijpen bij concretisering in het algemeen essentieel is, terwijl dat bij compilatie niet het geval is.

Anders dan bij programma's, ligt de nadruk bij specificaties niet op executeerbaarheid (en zeker niet op efficiënte executeerbaarheid), maar op formele hanteerbaarheid en mathematische helderheid. Om die reden hebben specificatietalen een overwegend *declaratief* karakter, dit in tegenstelling tot het overwegend *procedurele* karakter van programmeertalen. Niettemin is er geen scherp onderscheid tussen beide soorten talen. Het is zelfs wenselijk talen te ontwikkelen die beide functies kunnen vervullen, zodat de hele weg van specificatie naar implementatie binnen een en hetzelfde kader kan verlopen. En, al is executeerbaarheid van specificaties geen primaire eis, het kan grote voordelen hebben. De programmeur is dan immers in staat reeds in een vroeg stadium van het ontwerp met weliswaar zeer trage, maar niettemin werkende prototypes van het uiteindelijke systeem ervaring op te doen.

In overeenstemming met de aan specificaties gestelde eis van helderheid hebben specificatietalen in toenemende mate een *logisch/mathematisch* karakter. *Algebraïsche specificaties*, bijvoorbeeld, zijn axiomastelsels in de *logica van vergelijkingen*. Bij concretisering van algebraïsche specificaties moet men in eerste instantie denken aan *herschrijfsystemen*. Daarnaast wordt de *eerste-orde logica* als specificatietaal gebruikt. In dat geval ligt concretisering in termen van *Prolog* voor de hand, omdat deze programmeertaal gebaseerd is op een executeerbaar onderdeel van de eerste-orde logica. Verder spelen stelsels *recursieve vergelijkingen (denotationele semantiek)* en *de abstracte theorie van processen* een belangrijke rol. Deze laatste komt naar voren bij de specificatie van parallelle berekeningen, die toenemende aandacht krijgt als gevolg van de snel stijgende populariteit van computernetwerken enerzijds en ultra-parallelle berekeningen in very large scale integrated circuits anderzijds. In de toekomst zullen ongetwijfeld ook vormen van hogere-orde logica als specificatiemiddel hun intrede doen.

De informatici binnen het CWI zijn op elk van deze terreinen actief. Tevens wordt samengewerkt met Philips Natuurkundig Laboratorium, het Laboratoire de Marcoussis van de Franse telecommunicatie-industrie CIT

Alcatel, de Philips Research Laboratories in Brussel en het software house C.O.P.S. Ltd. in Dublin aan een algemeen software-ontwikkelingsmodel ten behoeve van de productie van hoogwaardige programmatuur zoals bijvoorbeeld vereist ten behoeve van telefooncentrales. De nadruk ligt daarbij op het toepasbaar maken van de zoëven genoemde logisch/mathematische specificatiemethoden. De samenwerking vindt plaats in het kader van het European Strategic Program for Research in Information Technology (ESPRIT), dat ten doel heeft de Europese informatietechnologie-industrie in de komende 10 jaar internationaal een betere concurrentiepositie te verschaffen.

Het is niet voldoende nieuwe methodologieën en specificatietalen te ontwikkelen. Ze moeten ook ter beschikking van de programmeur komen in de vorm van *ontwerp- en programmeeromgevingen*. Evenmin als er een scherp onderscheid is tussen specificatie- en programmeertalen, is er een duidelijke grens tussen omgevingen die alleen het programmeren in een bepaalde programmeertaal ondersteunen en omgevingen die het hele ontwerpproces ondersteunen. Programmeeromgevingen zullen in toenemende mate de taken van bedrijfssystemen gaan overnemen. Dit houdt verband met het feit, dat bedrijfssystemen steeds meer gezien worden als de implementatie van een of meer talen.

Ook op het terrein van programmeeromgevingen is het CWI actief. Ten eerste wordt er gewerkt aan een programmeeromgeving voor de conversationele taal B, die wegens haar gebruikersvriendelijkheid aantrekkelijk is voor de snel groeiende groep serieuze gebruikers van persoonlijke systemen. Ten tweede wordt er gewerkt aan een taalonafhankelijke programmeeromgeving, die steun biedt bij het programmeren in willekeurige talen. Een dergelijke omgeving is ook te beschouwen als een *generator*, die op basis van een taaldefinitie een programmeeromgeving voor die taal oplevert. Vanuit dit project wordt een toenemend beroep gedaan op de aanwezige theoretische kennis op het gebied van algebraïsche specificaties en herschrijfsystemen.

Op de wat langere termijn zijn twee belangrijke theoretische ontwikkelingen binnen de Programmeertechnologie te verwachten. Ten eerste, de opkomst van een discipline die men *Algoritmiek* zou kunnen noemen. De Algoritmiek beschouwt specificaties en programma's als (grote) formules en bestudeert programmamanipulatie, zoals de wiskunde de manipulatie van formules bestudeert. Dit ter ondersteuning van de gang van specificatie naar implementatie door middel van correctheidbehoudende transformaties zoals boven beschreven. Een tweede ontwikkeling die te verwachten valt is de opkomst van een mathematisch gefundeerde *structurele-complexiteitstheorie*, die theoretisch inzicht verschaft in het begrip complexiteit zoals dat in de Programmeertechnologie naar voren komt en die de tegenhanger vormt van de in de afgelopen 20 jaar tot ontwikkeling gekomen theorie van de complexiteit van berekeningen.

november 1983 Jan Heering