DIESREDE 1974

30 jaar informatica; Hoe zal het verder gaan?

REDE UITGESPROKEN TER GELEGENHEID VAN DE 13E DIES NATALIS VAN DE TECHNISCHE HOGESCHOOL TWENTE DOOR PROF. DR. IR. A.J.W. DUIJVESTIJN OP 29 NOVEMBER 1974

Zeer Gewaardeerde Toehoorders

Bij het herdenken van de dies natalis van een jonge hogeschool als de T.H. Twente gaan de gedachten onwillekeurig uit naar de ontwikkeling van de mens. In de eerste drie levensjaren zijn de ouders bij nacht en ontij bezig het kind te verzorgen. Intussen is de kleuter gaan praten, waardoor een nieuwe vorm van communicatie ontstond. Aanvankelijk is het kind meegaand en leergierig. Het ontwikkelt zich snel en voorspoedig. Maar als het zo'n 13 jaar is geworden beginnen de eerste puberteitsverschijnselen zich te openbaren.
Aan de T.H. Twente zijn we met ons onderwijs in het 11de cursusjaar gekomen. Om aan de wensen van het moeilijker wordende kind tegemoet te komen hebben we onze organisatie geherstructureerd en op het ogenblik zijn we druk doende ons onderwijs op zijn kop te zetten met het doel dat te verbeteren. Het vakgebied van de informatica, waarover ik U vanmiddag iets zal vertellen, is ook nog jong. Ik heb het jaar 1944 als startpunt genomen. De informatica is dus in haar 30ste levensjaar. In de voorafgaande tijd is de informatica snel gegroeid. Problemen zijn ook haar niet bespaard gebleven. Ouderdomsverschijnselen zijn nog niet waar te nemen. De informatica is nog springlevend.

Algemeen wordt de elektronische rekenmachine tot de ontdekkingen gerekend die het maatschappelijk leven op beslissende wijze hebben beïnvloed. De ontwikkeling van de rekenmachine is in de laatste wereldoorlog in de VS begonnen. Men wilde daar sneller dan voorheen en zonder fouten ballistische tabellen berekenen. Eén van de eerste computers werd in 1944 gemaakt door Howard H. Aiken te Harvard. De machine die de Mark 1 werd genoemd, was nog gemaakt met elektromechanische componenten. In 1946 werd de eerste elektronische rekenmachine, ENIAC, door Eckert en Mauckley voltooid. Daarbij werd gebruik gemaakt van elektronenbuizen. In 1945 stelde John von Neumann voor, het programma te coderen en evenals de gegevens in het geheugen op te nemen. Hiermee was de elektronische machine volgens het zg. *stored program* concept geboren. Het idee van von Neumann en de intrede van de elektronica hebben de ontwikkeling en toepassing van computers in een stroomversnelling gebracht.
Al in 1951 kwam Univac met een computer op de markt. Deze machine was uitgerust met een trommelgeheugen. Intussen was in 1947 door Bardeen en Brittain de puntcontacttransistor en in 1949 door Shockley de lagentransistor uitgevonden. Al in 1955 werd de rol van de elektronenbuis in de computer overgenomen door de veel betrouwbaarder transistor. De uitvinding van het magnetische kerngeheugen in 1951 door Forrester, destijds nog student aan M.I.T., heeft het trommelgeheugen verdrongen als primair geheugen. De eerste machines, uitgerust met een kerngeheugen van enige omvang kwamen omstreeks 1957 op de markt. In 10 jaar tijd is de omvang van het kerngeheugen met een factor 10 toegenomen. Tegenwoordig werkt men met kerngeheugens die 1 à 2 Mbytes bevatten.
In het grensgebied van de apparatuur en programmatuur zijn twee vindingen van grote betekenis geweest. De eerste is de vinding van het subroutinemechanisme dat voor het eerst in de EDSAC machine in Cambridge in Engeland werd toegepast. Hiermee kon veel geheugenruimte bespaard worden. De tweede is de ontdekking van het ingreepmechanisme, omstreeks 1958-1959, waardoor veel efficiënter met de randapparatuur kon worden samengewerkt. Dat hiermee ook een aantal enorm moeilijke problemen werd opgeroepen kon men toen nog niet voorzien. Het is overigens onze landgenoot Dijkstra geweest die ons de gereedschappen en de theorie heeft verschaft om het probleem van de asynchrone parallelle processen de baas te kunnen.

De programmatuur heeft ook een enorme ontwikkeling doorgemaakt. In de begintijd werden de machines nog geprogrammeerd door bitrijen op te schrijven. Al spoedig hoefde dit niet meer toen de assembleertalen hun intrede deden. Om echter aan de automatiseringsbehoeften van de maatschappij te kunnen voldoen was al gauw duidelijk dat in de programmatuur een revolutionaire stap gedaan moest worden. Tijdens een discussie gedurende mijn verblijf aan het research laboratorium van IBM in Poughkeepsie in de winter 1957-1958 becijferden we globaal dat wanneer we zouden doorgaan alle programma's in assembleertalen te beschrijven, binnen 10 jaar alle Amerikanen het beroep van programmeur zouden moeten hebben. Het is dan ook niet toevallig dat in die tijd de eerste FORTRAN vertaler door John Backus en zijn team gemaakt werd voor de IBM-704. Vanaf dat moment zijn de computertoepassingen pas goed van de grond gekomen. Belangrijke programmeertalen die later werden gedefinieerd en waarvoor vertalers werden geschreven zijn ALGOL 60, COBOL, ALGOL 68, PL/1 en PASCAL. Tegenwoordig bestaan er al een kleine 1000 programmeertalen; waarlijk een Babylonische spraakverwarring. Op het ogenblik is het merendeel van de programmatuur in FORTRAN en COBOL geschreven. In verschillende opzichten echter zijn deze talen niet zo ideaal

gebleken. De programma-investeringen die in deze talen gemaakt zijn, zijn zo groot, dat het voorlopig niet duidelijk is hoe we deze talen ooit over boord kunnen zetten als er betere talen zouden worden uitgedacht.
Ook het gebruik van de computer is de laatste 30 jaar sterk veranderd. Aanvankelijk had iedere programmeur de machine voor zich alleen om zijn programma uit te testen. Doordat de dure apparatuur erg inefficiënt werd gebruikt is naar mogelijkheden gezocht hierin verbetering te brengen. Omstreeks 1956 kwam een *batch operating systeem* ter beschikking, dat ervoor zorgde dat de machine automatisch overging op een volgend karwei. Hiermede werd al een behoorlijke verbetering bereikt. De programmeur werd gedwongen achter zijn bureau een plan te maken voordat hij een testrun deed. Toch werd de apparatuur nog onvoldoende benut vooral omdat tijdens het transport van invoer respectievelijk uitvoer de machine stond te wachten. Een belangrijke vooruitgang werd geboekt toen men omstreeks de 60er jaren gebruik ging maken van multiprogrammering. Op dit thema is lange tijd gevarieerd. Op het einde van de jaren 60 kwamen de *time sharing systemen* in zwang.

Schrijfmachine en beeldbuis hebben als eindstation al een belangrijke plaats in de computerwereld veroverd. Het verzamelen van gegevens op afstand en het interactief rekenen hebben al een voorname plaats ingenomen.
De *operating systemen* die zorg dragen voor de verkeersregeling in de multiprogrammerings- en time sharing systemen zijn steeds omvangrijker en gecompliceerder geworden. Zonder te overdrijven mogen we stellen dat het merendeel van de systemen die zijn ontworpen en geïmplementeerd te ambitieus waren, afgemeten aan de kennis en het begrip dat de makers hadden van een theorie die aan deze systemen ten grondslag zou moeten liggen. De resultaten waren er dan ook naar. Monsterachtige systemen, waarin veel fouten bleven zitten, waren het gevolg en ook de overhead die van de machine werd gevraagd was groot. Het is gebleken dat de bouw van grote en gecompliceerde systemen een aparte aanpak vraagt. Om te beginnen is het uitermate onverstandig om systemen te bouwen die men intellectueel niet aankan. Vooral Dijkstra heeft in diverse voordrachten en publicaties stelling genomen tegen deze werkwijze. Hij heeft vooral gewezen op de noodzaak gestructureerde systemen te bouwen en van elke stap die in het ontwerp of de implementatie wordt gemaakt te bewijzen dat deze correct genomen is. Hiermede wordt bedoeld dat men verstandelijk tot de overtuiging moet kunnen komen dat de betreffende stap goed is geweest. Eventueel kan men daarbij de machine als hulpmiddel gebruiken. De structuur van het systeem zal daarom doorzichtig moeten zijn. We zullen daarbij niet meer pretenties mogen hebben dan we waar kunnen maken.
Hoe is de stand van zaken nu?
De behoefte aan computers is groot. In Nederland waren in 1972 al 1810 computers in gebruik, waarbij de minicomputers en de special-purpose machines niet zijn meegerekend.
In Amerika werd de eerste computer in 1951 geleverd. In 1962 waren er al 10.000 en in 1972 100.000. Men verwacht dat in 1975 dit aantal zal zijn verdubbeld. Na de automobiel- en olieindustrie komt de computer-industrie als derde in grootte. Volgens Diebold (1972) is de omzet in miljoenen dollars in de VS respectievelijk Europa in 1970 en zijn de schattingen voor 1975 en 1980 als volgt:

	1970	1975	1980
Europa	3900 M$	8100 M$	12850 M$
VS	10650 M$	22136 M$	36600 M$

De groeisnelheid in Europa bedraagt rond de 20%, in Japan zelfs 30%. In de VS is de groei wat gematigder maar toch nog altijd 15% per jaar. De computer wordt verreweg het meest gebruikt voor bedrijfs- en bestuurskundige toepassingen. Hiermee is bedoeld de geautomatiseerde informatieverzorging ten behoeve van overheid, industrie, handel en geldwezen. In de techniek, de natuurwetenschappen, de economie maar ook in de sociale wetenschappen, taalwetenschappen en de rechtswetenschappen worden computers gebruikt.
Het merendeel van de geautomatiseerde informatieverwerking is in COBOL geprogrammeerd. Er is een begin gemaakt met de *data base management systemen.* Verscheidene fabrikanten komen met pakketten hiervoor op de markt. Een aantal is gebaseerd op het *DBTG proposal.* Hierin zijn echter nog tal van onduidelijke en slecht geformaliseerde begrippen. Ook voor andere dan op het DBTG proposal gebaseerde systemen geldt, dat het logische niveau en het accespad-niveau niet gescheiden zijn. Hierdoor zijn in de toekomst, als er nieuwe apparatuur komt, grote conversies te vrezen. Met de data base management systemen dreigt hetzelfde te gebeuren als wat we hebben ondervonden met de programmeertalen COBOL en FORTRAN, n.1. dat ze te vroeg tot feitelijke standaard werden verheven, voordat men eigenlijk begrepen had wat programmeren was en hoe een programmeertaal er uit zou moeten zien. Het onderzoek in de data base systemen is nog in volle gang. Met name noemen we het *relationele data base model* van Codd. Hieraan wordt ook aan onze T.H. onderzoek gedaan door de groep Informatiesystemen van de vakgroep Informatica. De behoefte aan data base systemen is groot. De vraag is echter of het verstand het van de commercie zal winnen.

Bij het gebruik van computers voor technisch-wetenschappelijke toepassingen heeft men tot op zekere hoogte geleerd modellen te formaliseren en deze op de computer af te beelden. We dreigen echter op alle gebieden waar programmatuur gemaakt wordt vast te lopen op een aantal ernstige problemen. In 1955 waren de apparatuurkosten nog vier keer zo groot als die van de programmatuur. In 1974 ziet het er heel anders uit. Nu verhouden de apparatuurkosten zich tot die van de programmatuur als 30:70. Men verwacht dat dit in 1985 zal zijn opgelopen tot 10:90. De loonexplosie die in 1964 begon is daaraan niet vreemd. Daar programmatuurkosten voor het grootste deel

loonkosten zijn, doet zich reeds geruime tijd de noodzaak voelen deze kosten te drukken. Vandaar dat de computerindustrie en ook de gebruikers pogen uit te vinden of gestructureerd programmeren uitkomst kan bieden.
Waar liggen de problemen?
Een ernstig euvel is dat men steeds weer alle programmatuur opnieuw maakt. Als een computerfabrikant een nieuwe machine uitbrengt die niet dezelfde instructies heeft als zijn oude machine, wordt vrijwel steeds alle programmatuur, zoals vertalers en operating systemen volledig opnieuw gemaakt.
Het gebruiken van geconserveerde kennis zoals dat in de wiskunde gebeurt, waar men zich vaak op stellingen beroept, wordt in de informatica nog maar weinig toegepast.
Bij de overgang op een nieuw computersysteem zijn de kosten van het omprogrammeren gigantisch gebleken. Vele bedrijven die deze kosten expliciet hebben gemaakt, hebben daardoor besloten niet van computerleverancier te veranderen. Voorwaar een slecht soort klantenbinding.

Een ander probleem is dat het ontwerpen en realiseren van nieuwe programmatuur zo veel tijd kost. Verder bevat het product gewoonlijk veel fouten waardoor het testen lang duurt. Maar het vervelende is dat het niet mogelijk is de tijd, die nodig is om het programmatuurproject foutenvrij te krijgen, te plannen zodat deze projecten vaak onvoorspelbaar lang uitlopen.
Het aanbrengen van wijzigingen en uitbreidingen is eveneens een kostbare aangelegenheid als daarmee niet te voren rekening is gehouden. Onderhoudskosten van de programmatuur behoren daarom eveneens tot de problemen.
Is er wat aan te doen?

Naar mijn mening móet er een oplossing voor deze problemen komen, ook al omdat we in de toekomst niet genoeg mankracht zullen hebben om alle programmatuur te maken.
Om te beginnen zal er beter moeten worden geprogrammeerd. Programmeren is een moeilijk vak dat niet met een stoomcursus kan worden geleerd. Van meet af aan zullen die aspecten de aandacht dienen te krijgen die een zo groot mogelijke doorzichtigheid waarborgen. Relaties tussen grootheden die de correcte werking van een programma beheersen dienen, voordat het programma beschreven wordt, bewust gemaakt te worden.
Zij behoren mede tot het model dat op de computer afgebeeld moet worden.
Bij de implementatie zal men zich moeten bedienen van bekende datastructuren en eenvoudige programmastructuren, die plezierige correctheidseigenschappen hebben. Het gebruik van het proceduremechanisme neemt daarbij een belangrijke plaats in. Het procedure-mechanisme is immers bij uitstek het middel om een verband te definiëren tussen in- en uitvoervariabelen. Wil men de mogelijkheden van dit mechanisme ten volle benutten dan moet aan de eis voldaan zijn dat de procedure gemaakt kan worden zonder enige wetenschap omtrent de omgeving waarin de procedure gebruikt gaat worden. De procedure moet daarom te maken zijn op grond van de actie die de invoer als het ware in de uitvoer moet overvoeren. Dat betekent o.a. dat de globale variabele aan banden gelegd moet worden. Het betekent ook dat het verlaten van de procedure op een ordelijke manier dient te gebeuren. Na beëindiging dient te worden teruggekeerd naar die plaats in het programma die direct volgt op de plaats waar de procedure werd aangeroepen. Via het parameter-mechanisme communiceert de procedure met de buitenwereld.
Men spreekt soms wel van de modulaire opbouw. Bovenstaande opmerkingen hadden te maken met de wijze waarop geprogrammeerd moet worden. Er moeten echter ook betere gereedschappen komen. Vooral voor de grote projecten zoals vertalerbouw, de bouw van operating systemen, data base systemen en grote applicatiepakketten zijn de bestaande gereedschappen zoals FORTRAN, COBOL, ALGOL 60 en PL/1 onvoldoende.
Hoewel de eisen die aan goed programmeergereedschap gesteld moeten worden nog onvoldoende bekend zijn kan er toch wel iets over gezegd worden.

De taal zal een doorzichtige structuur dienen te hebben. Ondoorzichtigheid levert alleen maar fouten op. Fouten dienen overigens in een zo vroeg mogelijk stadium te worden bestreden. Daaruit volgt de eis dat zoveel mogelijk statisch door de vertaler gecontroleerd moet kunnen worden of de programmeur zich houdt aan de aanvaarde programmeerconcepten. Om een voorbeeld te noemen: het name concept van ALGOL 60 bij het gebruik van parameters is een slecht concept gebleken. Dichter bij het ideaal liggen het value concept, het result concept, een volledige type specificatie en de opgave van wat in- en uitvoer parameters zijn. De accesrechten vanuit procedures en programmamodulen naar buiten maar ook de accesrechten naar binnen dienen beter geregeld te worden. Ook de scope in verband met accesrechten zal nader onder de loupe genomen moeten worden.
Kan de statische controle niet worden bereikt dan is een dynamische controle een dwingende eis. In het verleden is deze eis vaak afgedaan met de opmerking dat de object programma's zo inefficiënt worden. Deze opmerking snijdt om twee redenen geen hout meer.
Ten eerste wordt apparatuur steeds goedkoper dan programmatuur zodat een beetje meer machinetijd niet zo'n ramp hoeft te betekenen. Ten tweede mogen we hopen dat met de komst van de microprogrammeerbare machines eindelijk de apparatuur zich bij de wensen van de programmatuur gaat aanpassen waardoor dit efficiëntieverlies nauwelijks meer optreedt. Het hoeft bv. weinig extra tijd te vergen om te controleren of een array buiten zijn grenzen terecht gekomen is als dit in het microprogramma gebeurt.

Machines zijn in het verleden uitsluitend ontworpen door mensen uit de hardware sector. De programmeurs hebben dat misschien aan zich zelf te wijten omdat zij het tempo van de technische ontwikkelingen niet konden bijbenen. Toch geloof ik dat ook het management van computerfabrieken hier schuld treft. Al in een veel eerder stadium had de opdracht tot het ontwerp van nieuwe machines gegeven moeten worden aan een groep waarin de programmatuur even zwaar was vertegenwoordigd als de apparatuur.

Een van de eerste fabrikanten die het machine-ontwerp aan de eisen van de programmeertalen heeft aangepast is Burroughs geweest met zijn B5000 en B6000 serie. Overigens is het commerciële succes nog relatief gering geweest. Dit is hoofdzakelijk te wijten aan niet rationele koopgewoonten van gebruikers. Met de micro programmeerbare machine B1700 is Burroughs tevens de eerste geweest die met het zg. S-talen concept geschiktere objectinstructies heeft geïmplementeerd ten behoeve van de gangbare hogere programmeertalen. Een volgende dringende eis die aan de programmatuur gesteld moet worden is die van portabiliteit. Portabiliteit bij programmatuur is, dat deze met betrekkelijk weinig moeite van de ene machine op een andere overgezet kan worden. In principe zijn er de volgende mogelijkheden om portabiliteit te bereiken. Men kan de programmatuur in een hogere taal beschrijven. Is op de nieuwe machine een vertaler voor die taal beschikbaar dan lijkt het probleem opgelost. In de praktijk blijkt echter dat de vertalers voor bv. FORTRAN, ALGOL en COBOL vaak niet onderling compatibel zijn. De beschrijving van vertalers en operatingsystemen levert in de zojuist genoemde talen ernstige problemen op. Dit kan komen omdat de juiste data-types ontbreken of omdat het moeilijk is typische machine eigenschappen te beschrijven. Het gevolg kan zijn dat de hoeveelheid gegenereerde objectcode overmatig groot is of dat geen efficiënte objectprogramma's worden verkregen. Men kan zijn toevlucht nemen tot de beschrijving van de vertaler in een betere taal waarin wel geschikte datatypes aanwezig zijn en waarmee wel machine afhankelijke eigenschappen beschreven kunnen worden.
Voorbeelden van dergelijke talen zijn BLISS, SPL, BCPL en TAAL. De laatste taal is ontwikkeld door de Systeem Programmatuur Groep van de vakgroep Informatica van de T.H.T. Heeft men een vertaler voor TAAL op machine Ml en wil men overgaan op machine M2 en heeft men bovendien de TAAL-vertaler in zichzelf beschreven dan kan men een vertaler voor M2 op machine M 1 genereren. Wel zullen machine-afhankelijke delen moeten worden herschreven, uiteraard in TAAL. Op deze wijze kan men ook een betere versie van de TAAL vertaler die weer in TAAL beschreven is, met behulp van de oude versie genereren. Voor details verwijs ik naar een artikel van C.Bron: Machinetaal - Dode Taal, Informatie, jrg. 14 nr. 7/8, juli/augustus 1972, pag. 376-382.
Een belangrijk hulpmiddel is dat van de abstracte machine met een object-instructieset die aangepast is aan de eisen die de vertaler stelt. Gaat men over van machine M 1 op machine M2 dan hoeven alleen de routines of macro's die bij de object-instructieset van M2 behoren te worden herschreven. De taaltransformatie kan men via verschillende tussentalen laten lopen. Men kan bv. een standaard interface tussen de verschillende fasen van de vertaler definiëren zodat gemakkelijk een bepaalde fase door een andere vervangen kan worden. Bij de laatste techniek blijft het vanzelfsprekend nodig om de instructies van de abstracte machine in termen van instructies voor hetzij M1 of M2 in bv. TAAL te beschrijven. We zullen dus de tweede methode met de eerste methode moeten combineren.
Deze techniek kan ook worden gebruikt bij het ontwerp van operating systemen. Dit wordt gedaan bij het Terminal Operating System 45 project dat eveneens door de S.P.G. groep van de vakgroep Informatica wordt gemaakt.
Ook bij andere programmatuur zoals data base systemen en applicatie-pakketten kan men deze techniek toepassen.
Het is hard nodig, dat alle programmatuur in de toekomst aan de eis van portabiliteit voldoet.
Wie zal er voor zorgen dat dit gebeurt?
In ieder geval zullen de computerfabrikanten zelf ten behoeve van hun eigen machines portable software dienen te maken willen zij nog enige rentabiliteit halen. De computerfabrikanten hebben er waarschijnlijk weinig behoefte aan, hun systemen zodanig portable te maken dat het systeem gemakkelijk door de concurrent overgenomen kan worden. Jammer, want ik geloof dat dit een kortzichtig standpunt is. Een goede standaardisatie opent toch weer nieuwe markten.

Wat betreft het maken van applicatie-pakketten zou ik willen voorstellen dat bepaalde bedrijfstakken gezamenlijk een instituut financieren met het doel portable applicatie-pakketten te maken. Vanzelfsprekend kan men hierbij van de diensten van software-houses gebruik maken. Ook de overheid zou hier stimulerend kunnen optreden. Een goed voorbeeld van een dergelijk applicatie-pakket is Genesys: een systeem voor uniforme informatieverwerking in de techniek dat is beschreven door J.Blauwendraad en Th.H. Kayser (zie hiervoor De Ingenieur. jrg. 86, nr. 39, 16 september 1974). Met Genesys kunnen civieltechnische en werktuigkundige problemen worden aangepakt.
Tot slot wil ik nog een trend noemen die een belangrijke mogelijkheid vormt tot zowel portabiliteit als verhoging van ontwerp-snelheid van programmatuur.
Ik doel hier op het voor speciale doeleinden ontwikkelen van uitdrukkingsmiddelen om een probleem in te beschrijven. Zo'n uitdrukkingsmiddel (of zo U wilt een taal) dient aangepast te zijn aan de gebruiker, d.w.z. hij moet de taal gemakkelijk kunnen leren. Formeel moet de taal scherp gedefinieerd zijn. Probleem-beschrijvingen kunnen dan door generatoren en vertalers worden omgezet in programma's die door een automaat geïnterpreteerd kunnen worden. Vanzelfsprekend moeten de generatoren weer aan bovengenoemde eisen van portabiliteit voldoen. Het is van belang dat gebruikersgroepen deze mogelijkheden onderkennen. Zij zullen de activiteiten in hun eigen vakgebied zodanig expliciet dienen te maken dat deze geformaliseerd kunnen worden en geschikt gemaakt voor computerafbeelding.

Toekomstige ontwikkelingen

In het verleden is de programmatuurontwikkeling steeds achtergebleven bij de apparatuurontwikkeling. Het ziet er niet naar uit dat daarin spoedig verandering zal komen.

Er is een aantal technische vindingen die het gezicht van de informatica drastisch kunnen en zullen veranderen. Met de in de jaren 1958- 1961 ontwikkelde planaire technieken in de Bell laboratoria in de VS werd de geïntegreerde schakeling mogelijk. De integratietechnieken zijn sedertdien vervolmaakt. Tegenwoordig kan men een volledige processor onderbrengen in een ruimte ter grootte van een lucifersdoosje. De betrouwbaarheid van de schakelingen werd sterk verbeterd. Vooral erg belangrijk is, dat de kosten van deze technieken aanmerkelijk lager liggen dan die van conventionele technieken. Men behoeft alleen maar te kijken naar de prijsontwikkeling van de *pocket calculators.*

We kunnen verwachten dat deze technieken in de eerstvolgende jaren ook bij minicomputers toegepast gaan worden. De prijzen van minicomputers zullen in de toekomst dan waarschijnlijk sterk omlaag gaan. Daar komt nog bij dat de onaantastbaar lijkende positie van het kerngeheugen werkelijk gaat afbrokkelen en dat zijn rol wordt overgenomen door I.C. (integrated circuitry) geheugens. Bovendien zijn de tegenwoordige minicomputers al vaak uitgerust met een FORTRAN of een COBOL vertaler. Dit alles bij elkaar genomen zou best tot gevolg kunnen hebben dat de verkoopsprijs van de minicomputers zodanig komt te liggen dat de bezettingsgraad voor de gebruiker veel minder van belang is. Het kan betekenen dat de positie van de grote multiprogrammeringscomputer aangetast gaat worden of dat de groei van nieuwe toepassingen niet bij deze grote computersystemen terechtkomt. Dat wil niet zeggen dat er in de toekomst geen grote systemen zullen bestaan. Er blijven altijd systemen nodig om de zeer omvangrijke problemen op te lossen, die veel rekentijd en/of geheugen vragen. Een factor die sterk kostenverhogend werkt bij de grote systemen is de complexe programmatuur die men tot nu toe nodig heeft gehad om ten eerste een omgeving te creëren waarbij iedere gebruiker als het ware de beschikking heeft over zijn eigen machine, ten tweede iedere gebruiker de illusie te geven dat hij over een groot lineair geheugen beschikt en ten derde om een soepele samenwerking te verzorgen met de randapparatuur die de gebruiker schijnbaar tot zijn exclusief bezit mag rekenen. Helaas hebben de bouwers van de operating systemen zich niet gehouden (of hebben ze er zich niet aan kunnen houden omdat zij deze technieken nog niet beheersten) aan het stapsgewijs introduceren van abstracties waardoor virtuele machines ontstaan die op een eenvoudige wijze in elkaar kunnen worden uitgedrukt en waardoor een veel overzichtelijker en betrouwbaarder systeem had kunnen ontstaan. De opmars van de minicomputer en van zelfs krachtiger machines zou de behoefte aan de virtuele processoren, die de grote computer ter beschikking stelt, kunnen overnemen.

Een ander idee waar we iets van zouden kunnen verwachten maakt gebruik van een principe dat ik zou willen aanduiden met *autoprincipe.* Een auto kan bediend worden door iemand die niet op de hoogte is met de technische verschijnselen die zich binnen de auto afspelen. De programmering van de auto is uiterst eenvoudig. De gebruiker heeft slechts te maken met het stuur, het gaspedaal, de koppeling, de versnellingshandel en nog een paar eenvoudige knoppen en schakelaars. Hij heeft zich geabstraheerd van de techniek en spreekt het apparaat toe door middel van een gebruikerstaal. Het gebruik van de pocketcalculator gaat volgens het autoprincipe. Het is te verwachten dat bij de opmars van de minicomputer ook een auto-effect zal optreden. Het zal daarbij niet alleen bij auto blijven: er zal dan een veelsoortige rij van eenvoudig te gebruiken toepassingen kunnen verschijnen, b.v. het verzorgen van de zoekfunctie, mutatiefunctie e.d. op grote bestanden of een systeem bestaande uit een verzameling terminals waarop men interactief in APL kan werken.

Een interessante veelbelovende ontwikkeling is de geheel in I.C. uitgevoerde microprogrammeerbare microprocessor. Hiermee kunnen digitale schakelingen gemaakt worden zonder te hoeven solderen. Het maken van de digitale schakelingen komt dan neer op het formuleren in een taal van de gewenste functie. Door middel van een vertaler kan deze functie automatisch worden omgezet in een bitrij die in het geheugen van de microprocessor wordt geladen, waarmee de digitale schakeling een feit is. Als we het kunnen bedenken en onder woorden kunnen brengen is het ook gemaakt. Dergelijke programmeerbare micro-processoren zijn al voor lage prijzen op de markt verschenen. Willen we in de toekomst meer van deze mogelijkheden gebruik maken dan zullen we de intelligentie-vragende delen van een systeem moeten opsporen en moeten scheiden van de energie-vragende stukken. Ik zal aan de hand van een voorbeeld, waarbij dit al gebeurt, uitleggen wat ik bedoel. We beschouwen een regeldrukker die gebruik maakt van een continu ronddraaiende ketting waarop alle mogelijk te drukken tekens éénmaal voorkomen. Als we bv. op kolom 41 een P afgedrukt willen hebben, bestaat de intelligentie hieruit dat wordt bijgehouden wanneer de P kolom 41 passeert. Van ieder teken moet worden bijgehouden waar het zich bevindt en op welke plaatsen een hamertje dat bij een kolom hoort geactiveerd moet worden. De besturing van de kettingdrukker gebeurt met behulp van digitale schakelingen. Vanzelfsprekend zal men de digitale signalen soms moeten omzetten in analoge signalen en moeten versterken om de benodigde energie te leveren voor het activeren van het hamertje. Om bovengenoemd principe uit te buiten zullen veel analoge signalen door digitale vervangen moeten worden. Zo zou men er bijvoorbeeld aan kunnen denken om in de televisietechniek de intelligentie die nodig is om de electronenstraal bepaalde wegen met een bepaalde intensiteit te laten doorlopen met digitale middelen uit te voeren.

Als het I.C.-geheugen het kerngeheugen inderdaad verdringt en vooral wanneer de geheugenomvang, die tegenwoordig voor de grotere systemen rond de 1 à 2 Mbytes ligt, dan een factor 10 of misschien wel 100 groter zou kunnen worden zal dat de adresseringsprogrammatuur sterk kunnen beïnvloeden. Deze zal dan vereenvoudigd kunnen worden.

Tenslotte noem ik nog een gebied dat door de 1.C. beïnvloedt zal worden. Men kan het feit dat de I.C. goedkoop is en weinig ruimte in beslag neemt, uitbuiten voor het parallelliseren van bewerkingen. Men moet daarbij denken aan parallelle polynoomverwerking (met 2 vermenigvuldigers en optellers kan men een polynoom van de graad p in ½p + log p - 2 vermenigvuldigingen uitvoeren). Evenzo kan men versnellingen bereiken met sorteren (ten hoogste n verwisselingen), matrixinversie etc. We kunnen derhalve verwachten dat daar, waar rekensnelheid van belang is ingewikkelde functies in I.C.s worden geïmplementeerd.

Naast de geïntegreerde schakelingen zijn er technische ontwikkelingen die de mogelijkheid in zich dragen, om aan de behoefte van opslagruimte te voldoen. De belangrijksten daaronder zijn het *magnetische bubble geheugen* en het *magneto-optische geheugen.* Met een magneto-optisch geheugen kan men grote schrijf-dichtheden bereiken. De toegangstijd tot de informatie is vele malen kleiner dan die van het schijvengeheugen. De geheugenomvang daarentegen is veel groter. Het magneto-optisch geheugen is het beste te vergelijken met een zeer groot kerngeheugen. Als een dergelijk apparaat ter beschikking zou komen zal de huidige kennis van de datastructuren en bestandsorganisaties, die voor een groot deel doorspekt zijn met apparaatafhankelijke eigenschappen, opeens volstrekt verouderd zijn.
Vandaar dat de eerdergenoemde ontwikkeling van abstracte systemen zoals het relationele data base model van groot belang geacht moet worden voor de toekomst.

Het bubble geheugen is het beste te vergelijken met een zeer groot aantal schuifregisters. Het heeft uiterst ongewone eigenschappen waarmee we in systemen nog geen enkele ervaring hebben. Het gebruik in systemen zal dus van de grond af moeten worden opgebouwd. Wel is het zeker dat er zeer grote geheugens mee gemaakt kunnen worden, ook weer met een geringe toegangstijd. Het is mogelijk uitvoeringsvormen te bedenken die op een groot kerngeheugen lijken. Of deze geheugens bruikbaar zullen blijken zal de toekomst moeten leren. Er moet nog veel ontwikkelingswerk worden gedaan op dit gebied. Datzelfde geldt ook voor de zg. *holografische geheugens,* waarmee complete pagina's tekst op een zeer kleine oppervlakte opgeslagen kunnen worden.
Het is wel duidelijk dat alle bovenbeschreven ontwikkelingen de opgedane kennis in de informatica snel kunnen doen verouderen. Bij de opleiding van informatici zullen we daar terdege rekening mee moeten houden.
In de software kunnen we helaas niet dezelfde spectaculaire toekomstige ontwikkelingen aanwijzen als in de hardware, hoewel de software-ontwikkeling in allerlei opzichten niet stilgestaan heeft. De vertalerbouw hebben we zo langzamerhand onder de knie. Van het vele werk dat wordt gedaan aan correctheidsbewijzen hebben we zeker het een en ander geleerd. De verwachting die Diebold (research rapport 1974) uitspreekt dat in de jaren 1980-1985 correctheidsverificatoren zullen bestaan moet als onrealistisch worden beschouwd.
Wel is in de software nog een aantal interessante doelen aan te geven zoals:

- Het bezinnen op de vraag hoe men moet programmeren en hoe men systemen moet bouwen;
- Het ontwikkelen van goede programmeergereedschappen, waarbij correctheid, overzichtelijkheid en foutendetectie voorop moeten staan;
- Het ontwikkelen van abstracte modellen van gegevensbanken, datacommunicatienetwerken, computernetwerken en operating systemen;
- Het vergroten van de betrouwbaarheid en beschikbaarheid van informatieverwerkende systemen;
- Het formaliseren van toepassingen in allerlei gebieden van techniek, bedrijfs- en bestuurskunde etc.

De maatschappij stelt haar eigen eisen aan de informatica. Zij wil het probleem van de veelheid van informatie opgelost zien; zij heeft behoefte aan selectieve informatie. Ook de opslag van vertrouwelijke informatie zal bevredigend geregeld moeten worden. Als bovendien de opwaartse beweging van de prijsontwikkeling van gedrukte informatie zich in de toekomst voortzet zal behoefte ontstaan aan andere vormen van informatieverspreiding. We moeten denken aan een opvolger van het T.V.-toestel, die de toegang tot gegevensbanken mogelijk maakt en zekere interactie met een openbaar systeem opent. Daartoe zullen de nodige communicatienetwerken opgebouwd moeten worden. Het is van belang bij de discussie over kabeltelevisie deze toekomstige vraag in de overweging te betrekken.
Opleiding tot informaticus
De belangrijkste vraag is: Hoeveel informatici heeft de maatschappij nodig? Er zijn ruwweg drie methoden om hier iets over te zeggen. De eerste methode gaat uit van het aantal computers dat geplaatst is. Men bepaalt het gemiddelde van het aantal functionarissen die met de computer moeten werken. Een bepaald percentage daarvan is academisch opgeleid. Verder neemt men een bepaalde groei van het computerbestand aan.

geplaatste computers in Nederland		geschatte aantal computers in Nederland volgens (RCF)	
1968	650	1975	2590
1969	930	1976	2860
1970	1120	1977	3140
1971	1430	1978	3420
1972	1810	1979	3700
1973	2060	1980	3970
1974	2310		

De tweede manier maakt gebruik van enquêtering. De derde manier veronderstelt dat Europa dezelfde ontwikkeling zal doormaken als de VS met een vertraging van *5* jaar. De commissie Frielink (FR) heeft in 1968 volgens de eerste en derde methode gewerkt (Rapport van de Commissie Opleiding van deskundigen voor Automatische Informatieverwerking, het Nederlands Studiecentrum voor Administratieve Automatisering, augustus 1969).
De enquête Gebruik Computers in Nederland (GCN) blijkbaar volgens de tweede methode. (Het gebruik van computers in Nederland, Studiecentrum NOVI/Samson 1973).
Het artikel Raming van het aantal Computerfunctionarissen in 1980 in Nederland en België - Luxemburg door L. de Leeuw en G. Thiers (Informatie juli/augustus 1974) vermeldt niet welke methode is gebruikt (RCF).
De resultaten luiden als volgt:

Gemiddelde personeelsbehoefte per installatie

FR	10.4	waarvan	25% academici
GCN	12.7	waarvan	8.5 in het rekencentrum en 3.2 erbuiten 1 academicus per installatie
RCF	9.6	geen uitspraak over opleiding	

Neemt men RCF als de meest realistische schatting van het aantal computers en GCN als de betrouwbaarste schatting van het aantal academici per installatie dan zouden in 1974 300 academici nodig geweest zijn. De sectie informatica van de academische raad gebruikt als schatting 400 academici per jaar. Tussen haakjes merk ik op dat het aantal elektronici dat jaarlijks in Nederland afstudeert ongeveer 225 bedraagt.
In West-Duitsland heeft men de informatica-opleiding met het Zweites Datenverarbeitungsprogramm der Bundesregierung grondig aangepakt. Van de 35 universiteiten zijn er 15 die een afdeling informatica hebben en die opleiden voor Diplom-Informatiker. Met de methode van enquêtering heeft men berekend dat er in de Bondsrepubliek 1500 Diplom-Informatiker nodig zullen zijn op een bevolking van 60 miljoen. Tegen deze cijfers zouden in Nederland op een bevolking van 13 miljoen 325 academici nodig zijn. De Amerikaanse situatie is moeilijk vergelijkbaar met de onze omdat men daar met computerontwikkeling veel verder is. Men zou kunnen verwachten dat wij dezelfde ontwikkeling als de VS zullen doormaken. Vergeet echter niet dat in de VS het computerpark het sterkst is uitgebreid toen de informatica als discipline nog nauwelijks bestond.

De chaotische toestand die daardoor in de software ontstond heeft er toe geleid dat overmatig veel mensen werden ingeschakeld. Bovendien constateren we dat de vraag naar informatici in de VS afneemt. Hiervoor zijn misschien verschillende oorzaken aan te wijzen. Een ervan is dat de softwarekosten zeer sterk zijn toegenomen. Een ander is wellicht dat door de economische recessie, mede als gevolg van de energiecrisis, in een aantal overheidsprojecten, zoals het ruimtevaartprogramma, het mes is gezet. Een direct gevolg van de energiecrisis is de slechte gang van zaken in de automobiel- en vliegtuigindustrie. Deze industrieën hadden enorme aantallen informatici in dienst die met het grootste gemak aan de dijk werden gezet.

Uit economische noodzaak zal men zo snel mogelijk uit die softwarecrisis dienen te komen. In 1957 stapte men over van assembleertalen naar hogere programmeertalen. Een vergelijkbare stap lijkt nodig.
Deze moet worden gezocht in de richting van introductie van een niveau van beschrijvingen van toepassingen dat een trap hoger ligt dan dat van de huidige hogere programmeertalen. Met behulp van deze beschrijvingen moeten de programma's worden gegenereerd om deze interpreteerbaar te maken voor een automaat. Zoals al eerder betoogd zullen de daarvoor benodigde software en ook hardware beter gestructureerd dienen te worden en zullen de gereedschappen drastisch moeten worden verbeterd. Dit kan men niet met horden mensen bereiken. Hiervoor zijn kwalitatief goede mensen nodig. Ik geloof dat we de ontwikkelingen in de VS ter harte moeten nemen.

Ook in Duitsland blijkt de vraag naar informatici enigszins te zijn afgenomen. Ik ontken niet dat we met de automatisering van de informatieverwerking ten opzichte van de VS ver achter liggen, maar wat betreft het toepassen van computers slaan we in Europa geen gek figuur.

Met de opleiding van informatici ligt Nederland achter ten opzichte van Engeland en Duitsland. Daar moet natuurlijk verbetering in komen omdat wij steeds meer aangewezen zullen zijn op die gebieden waarvoor een intellect-intensieve inbreng vereist is.

Een en ander tegen elkaar afwegend ben ik geneigd te veronderstellen dat het aantal van 400 academisch opgeleide informatici wat aan de hoge kant ligt. Met een 250-tal komen we waarschijnlijk dichter bij de toekomstige behoefte. Overigens zal het nog jaren duren voordat dit aantal van 250 per jaar kan worden bereikt. In de vakgroep informatica zijn momenteel 18 ingenieurs opgeleid, in de vakgroep digitale techniek 13.
In de ingelopen toestand kunnen de beide vakgroepen ongeveer 35 ingenieurs per jaar afleveren.
In Delft werden tot nu toe in de informatica 72 ingenieurs opgeleid; van de schakeltechniek waren mij de cijfers niet bekend. Per jaar leverde Delft de laatste 4 jaren gemiddeld 12 informatica-ingenieurs af. De opleidingscapaciteit is naar schatting hoger. Waarschijnlijk worden in Nederland in totaal aan de instellingen van wetenschappelijk onderwijs op dit ogenblik niet meer dan 50 academici met als hoofdvak informatica opgeleid. Wel worden in andere richtingen afgestudeerden in het bedrijfsleven ingezet in de informatica zonder daartoe de geëigende vooropleiding te hebben gehad.

Het is bekend dat uitbreidende organisaties aan groeiwetten zijn gebonden wil de organisatie nog functioneren. Als men boven 30% groei per jaar komt wordt de gehele capaciteit van de organisatie gebruikt om de nieuw aangetrokken mensen in te werken. Nemen we dit maximale groeipercentage aan voor de studenten en de benodigde medewerkers, van wie er volgens het rapport van de sectie informatica van de academische raad circa 150 (toename ca. 95*)* moeten zijn, dan duurt het tenminste 6 jaar voordat we 250 informatici per jaar afleveren en 8 jaar voordat we er 400 per jaar afleveren. Om de wetenschappelijke medewerkers in te werken is minstens 4 jaar nodig. Vóór die tijd kan men deze WMs moeilijk inschakelen bij de begeleiding van afstudeerders.
Het moet dus uitgesloten worden geacht dat op de korte termijn, waarvan het rapport van de sectie informatica van de academische raad spreekt, deze aantallen wetenschappelijke medewerkers ingewerkt kunnen worden. Met het benodigde aantal hoogleraren is het nog somberder gesteld. Op korte termijn zijn 12 min./16 max. hoogleraren en lectoren nodig. Waar moeten we die vandaan halen? Het aantal deskundigen in de software engineering dat ook nog over bedrijfservaring beschikt is zeer klein.
In Twente is, sedert de start van de Hogeschool in 1964, het beleid in de afdeling der Electrotechniek er op gericht geweest de informatietechniek en de informatica uit te bouwen. Later is dat beleid mede door de onderafdeling der Toegepaste Wiskunde overgenomen. Ik ben van mening dat de informatica-groepen (en dat geldt voor alle 3 THs), die al een behoorlijke sterkte hebben opgebouwd niet in ontwikkeling moeten worden geremd; integendeel verder moeten worden uitgebouwd.

Aan wat voor academici in de informatica heeft de maatschappij behoefte? Als men onderscheid maakt naar hoofdzakelijk theoretisch opgeleiden, informatici voor systeemprogrammatuur en technisch-wetenschappelijke applicaties en bedrijfsinformatici, dan is naar mijn mening het aantal benodigde bedrijfsinformatici verreweg het grootst.
Het aantal theoretisch opgeleiden zal betrekkelijk klein dienen te zijn. Een redelijke verhouding tussen deze categorieën lijkt 1:3: 12. Bij een aantal van 250 informatici per jaar zou dat op 15, 45 resp. 190 per jaar neerkomen.

Wat de opleiding in de bedrijfsinformatica betreft zie ik twee stromen.
De eerste stroom, die we in Twente sinds een jaar of vier aan 't opbouwen zijn, is die van exact opgeleide academici (meestal wiskundig ingenieur) die de informatica als hoofdvak hebben gekozen en zich bovendien hebben bekwaamd in de bedrijfs- of bestuurskunde. We mogen ons verheugen in het feit dat veel wiskunde studenten deze maatschappijbehoefte hebben onderkend en deze opleidingsweg hebben gekozen. Dat deze maatschappijbehoefte er is, volgt duidelijk uit het feit dat ten behoeve van de bestuurlijke gegevensverwerking veel elektrotechnici en fysici door de bedrijven zijn omgeschoold. De tweede stroom van bedrijfsinformatici moet worden opgeleid daar waar afdelingen der bedrijfskunde, bestuurskunde of economie bestaan. Deze opgeleiden hebben hun hoofdaccent in de bedrijfskunde, bestuurskunde of economie én in de informatica. Zij zijn de aangewezen mensen voor de beroepen in de systeemanalyse. Het systeemontwerp en de systeemprogrammering zou ik overwegend aan de eerste stroom willen toevertrouwen, hoewel het systeemontwerp ook door de informatici uit de tweede stroom kan worden gedaan.

Zowel in de systeemprogrammatuur, de technisch-wetenschappelijke toepassingsprogrammatuur als in de bedrijfsinformatica is een ingenieursmentaliteit nodig om systemen te kunnen ontwerpen en te realiseren waarbij een kostenbesef vereist is en waarbij het binnen de gestelde tijd afleveren van een product niet ongewoon is. Ook in deze sectoren is management van groot belang. Dat laatste is in de informatica-opleiding nog een verwaarloosd gebied.
Ook ten aanzien van de opleidingen kunnen we van de VS leren. In een artikel in de Communications van de A.C.M. van juni 1972 stelt Kandel dat de Amerikaanse universiteiten een te grote nadruk leggen op de theoretische aspecten van de informatica zoals automatentheorie, booleaanse algebra, tralies, semigroepen en universele algebras.

Hij stelt: "Industry gets graduates from computer science departments with a bag full of the last technical jargon but no depth of understanding of real computer systems and with no concept of the problems they will be asked to face and solve. Computer science graduates have programmed only a limited set of problems in computer languages, many of which they have never finished or tested adequately. The average graduate student has no solid idea of what happens when design projects get large, as most real-life systems do".
Hij concludeert: "This can easily be accomplished by combining theoretical courses with a practical approach to computer engineering."
Ook in Duitsland klaagt de industrie over te veel uitsluitend theoretisch opgeleiden. Het gevaar daarvan is dat ze in de industrie geen plaats kunnen vinden en dan noodgedwongen in het onderwijs belanden.
Het is gewenst dat van leraren aan de HIO en BIO-scholen zoals dat ook aan de HTSen gebruikelijk is, geëist wordt dat zij op tenminste enige jaren praktijkervaring kunnen bogen.
De conclusie die ik hieraan wil verbinden is dat de systeemprogrammatuurdeskundigen, de technische toepassingsinformatici en de eerste stroom bedrijfsinformatici aan de T.H. ‘s moeten worden opgeleid. Het komt mij onjuist voor dat universiteiten die niet beschikken over opleidingservaring met ingenieurs een dergelijke opleiding zouden verzorgen.
De eerste stroom bedrijfsinformatici wordt in Twente al opgeleid, we hebben hier immers de beschikking over afdelingen bedrijfs- en bestuurskunde (i.o.).

Ook voor de tweede stroom bedrijfsinformatici is Twente bijzonder geschikt. De sterk theoretisch gerichte opleidingen zullen door de universiteiten verzorgd kunnen worden, maar moeten gering in aantal blijven.
Bij voorkeur dient een combinatie met een bedrijfsinformaticaopleiding te worden nagestreefd.
Tot slot nog iets over de opleiding zelf.
De opleiding moet wel praktisch georiënteerd zijn. Eigenlijk is het beter te zeggen: technisch georiënteerd. Ze dient zeker fundamenteel te zijn. Zoveel mogelijk ontdaan van franje die men beter in het bedrijfsleven erbij kan leren. De informaticaopleiding mag vooral niet te smal worden.
Het lijkt mij dat we hier in Twente principieel moeten vasthouden aan een combinatie van hetzij elektrotechniek, werktuigkunde of chemie, informatica en een wiskundige basis, hetzij wiskunde, informatica en een bijvak in de techniek, bedrijfs- of bestuurskunde, hetzij bedrijfs-, bestuurskunde of economie, informatica en wiskunde als bijvak.
De informatica is aan zoveel veranderingen onderhevig dat ik ten opzichte van de afgestudeerde niet de verantwoording op mij zou durven nemen hem met een smalle basis de maatschappij in te sturen.
Toch zal ten behoeve van maatschappelijke duidelijkheid voor de afgestudeerden met als hoofdvak informatica een aanduiding nodig zijn, hetzij door het instellen van een diploma informatica-ingenieur of -doctorandus, hetzij door het instellen van een studierichting of door het instellen van een afstudeerrichting binnen een afdeling.

Verder dienen we er rekening mee te houden dat bij harmonisatie binnen de EEG naamgeving aan de informaticadiploma's vereist zal zijn. Uit het bovenstaande moge blijken dat ik een sterk voorstander ben van verschillende opleidingswegen die zijn gekenmerkt door een zekere breedheid. Een zelfstandige informatica-afdeling biedt te weinig waarborg dat deze breedheid gehandhaafd blijft. Een interafdeling lijkt een veel betere organisatie om bovenstaande doelen te bereiken. In Twente zou er in feite weinig veranderen. Er is immers een interafdelingsvakgroep informatica van de afdelingen Elektrotechniek en Toegepaste Wiskunde. Bovendien is er de werkgroep bedrijfsinformatica, die bestaat uit leden van de vakgroep Economische Bedrijfskunde, de vakgroep Informatica en de vakgroep Operations Research. De verantwoordelijkheid voor de curricula kan men voorlopig leggen bij een contactgroep informatica. Met een soortgelijke groep hebben we goede ervaringen opgedaan toen de wiskundigen en natuurkundigen nog lid waren van de technische afdelingen.

In de Volkskrant van 12 september 1974 stond een stukje met de titel: Minder werk door computers. Dat dit een misleidende of eigenlijk dubbelzinnige stelling is blijkt uit de praktijk. Werkbesparing is uiteraard een van de motieven om een computer aan te schaffen in een bedrijf, maar het gevolg van de aanschaf is meestal dat er méér personeel moet komen. Veel taken, die met mensenhand onuitvoerbaar waren, kunnen door de computer wel worden aangepakt. Daarbij is ook veel routinewerk. Door de computer kan de mens zich met interessantere dingen dan routinearbeid bezighouden. De persoonlijke ontplooiing zou daardoor bevorderd kunnen worden. Het bovengenoemd artikel was gebaseerd op een rapport van twee medewerkers van het Centraal Planbureau (no. 2/ 1974, Investeringen, lonen, prijzen en arbeidsplaatsen, H. den Hartog en H.S. Tan). In dit rapport kon ik niets over computers vinden. Wel besprak de publicatie het verschijnsel dat sinds 1964 de investeringen in machines relatief sterker waren toegenomen dan daarvoor met het doel de factor arbeid te verminderen; mechanisering dus. Mechanisering kan tot op zekere hoogte menselijke arbeid vervangen, maar door mechanisering kan ook werk worden gedaan dat anders niet uitgevoerd had kunnen worden. Vaak ontstaat er dan nieuw werk waarvoor weer mensen nodig zijn.

Kreiken heeft dit jaar op de voorlichtingsdag gesproken over de z.g. *transferverschijnselen.*
De eerste, *transfer of energy* begon met de komst van de stoommachine. Daarbij werd menselijke spierkracht vervangen door kunstmatig opgewekte energie.
De tweede, *transfer of skill* begon met de industrialisatie, toen geschoolde arbeid door middel van bepaalde methoden, werktuigen en machines geschikt gemaakt werd voor ongeschoolde arbeiders.
De derde, *transfer of thought* (al is dit naar mijn smaak een wat wijdse naam) begon met de mechanisatie.

Eenvoudige arbeid werd door machines overgenomen (zie ook Principles of Industrial Organization, D.S. Kimball and D.S. Kimball Jr., pag. 18). De informatica houdt zich in wezen met de transfer of thought bezig. In dat gebied ligt veel werk te wachten. Sinds 1964 bevinden we ons weer in zo'n transferperiode. Het is goed in de geschiedenis terug te kijken naar wat er in zo'n transferperiode gebeurde. Er ontstonden sociale wantoestanden. De maatschappij verzette zich met het doel alles bij het oude te laten. Zou het niet beter zijn lering te trekken uit het verleden en te anticiperen op toekomstige ontwikkelingen door tijdige bijscholing van de werkende bevolking?
Er wordt erg veel gedaan om mensen om te scholen als ze werkloos zijn. Maar is het dan feitelijk al niet te laat?
Ik geloof dat we naar een werkelijke education permanente toe moeten die anticipeert op de toekomst. Dat betekent dat we voor het onderwijs offers moeten brengen. Het onderwijs is al veel te duur wordt gezegd en het leek erop dat het hele nationale inkomen binnenkort aan onderwijs gespendeerd zou gaan worden. Daarom zijn we de laatste jaren bezig geweest te proberen deze kosten te drukken. In het hoger onderwijs b.v. denkt men goedkoper uit te zijn dan voorheen, door een vierjarige opleiding in te voeren. Schattingen daarover roepen evenwel de nodige twijfels op. Wel zouden de afgestudeerden eerder in de maatschappij terecht kunnen komen. Maar moet dat nog wel, gezien de toenemende werkloosheid?
In het middelbaar onderwijs heeft men het vooral in schaalvergroting gezocht. Hebben we daarmee echter niet een aantal nieuwe problemen in huis gehaald? Iedereen mag het onderwijs volgen waarvoor hij geschikt is. Maar wie maakt dat uit?
Op de mammoetscholen is geen tijd meer voor persoonlijke aandacht. Alleen degenen die zich schikken naar de regels, die nodig zijn om zo'n grote kolos te laten functioneren doen het goed in zo'n systeem. Creativiteit wordt niet meer ontplooid. De motivatie is vaak ver te zoeken. Ook op de mammoetuniversiteiten zijn deze verschijnselen waar te nemen. Toch hebben we creativiteit broodnodig om de toekomstige problemen op te kunnen lossen. In de geherstructureerde universitaire opleiding moet er plaats blijven om die creativiteit te ontwikkelen. De studie mag niet gereduceerd worden tot het louter leren van een aantal vakjes. De student moet tijdens zijn studie ruim de kans krijgen een zelfstandig onderzoek te doen. Daarbij moet er gelegenheid zijn om voldoende kontakten te onderhouden met zijn wetenschappelijke omgeving. Voorwaarden voor een goed contact zijn tijd en persoonlijke inzet. Willen we profijt trekken van de mogelijkheden die de transfer of thought ons kan bieden dan zal de mens moeten leren zich persoonlijk te ontplooien. Daarvoor is een mentaliteitsverandering nodig. Om dit te bereiken zal het onderwijs wezenlijk moeten veranderen. We moeten de technische verworvenheden, waaronder zeker ook de computer, in dienst stellen van de mens en niet omgekeerd.