Poging tot plaatsbepaling van de Informatica.

Het doel van dit geschrift is in eerste instantie om voor mijzelf tot klaarheid te komen ten aanzien van de wetenschappelijke positie en pretentie van de Informatica -hoewel natuurlijk een ieder, die mijn conclusies delen wil, van harte welkom is-, omdat ik deze klaarheid als noodzakelijk ervaar, voordat ik zinvol over bv. een curriculum kan denken. Allerlei zakelijke en politieke overwegingen -aan de THE gegroeide situatie, verwachte kwaliteit van studenten, etc.- wil ik pas in tweede instantie in de beschouwing betrekken en vallen dan ook buiten het bestek van dit geschrift.

"A computer is a many-sided thing" en op grond van deze onontkoombare observatie komen geregeld voorstellen naar voren voor wat ik "een cocktailopleiding" zou willen noemen: een mengsel van bestaande ingredienten (electrotechniek, wiskunde, economie, organisatieleer, etc.etc.), waarvan men hoopt, dat het smakelijk is en, mits in voldoende mate genuttigd, aanleiding zal geven tot de gewenste wetenschappelijke feeststemming. Voor mijn gevoel ontstaat een dergelijk voorstel altijd "faute de mieux", dwz. doordat men voor de Informatica niet een specifiek eigen werkterrein met zijn eigen spelregels ontwaart, maw. doordat men niet het gevoel heeft, dat Informatica als een voldoend isoleerbare discipline bestaat. De bedoeling van dit geschrift is, om te onderzoeken in hoeverre een isoleerbare discipline, die ik dan gemakshalve "informatica" noem, onderscheidbaar is. Ter vermijding van misverstand, het gaat hier om isoleerbaarheid op grond van intrinsieke eigenschappen van diverse werkterreinen en niet om politieke isoleerbaarheid: ons gereedschap is dan ook niet de interdisciplinaire enquete, maar een zorgvuldige analyse.

Laat mij met een historisch overzichtje beginnen. Hoewel in het aanzwengelen van de programmeerbare rekenautomaat de wiskundige von Neumann een spectaculair aandeel heeft gehad, hoewel de eerste werkende -nou ja!- machine, waarin het programma in het geheugen opgeslagen was, de EDSAC, ontwikkeld is in the Mathematical Laboratory, hoewel in Nederland het Mathematisch Centrum snel volgde, waren de technische problemen om binnen de begrenzingen van budget en technologie een werkend apparaat te maken in de beginperiode allesoverheersend. (Dit wordt nog weerspiegeld in de toen gekozen naam van de professionele genootschappen zoals ACM, BCS en NRMG!) Van de technische wetenschappen, die hier in het geding geweest zijn, heeft de electronica in antwoord op de ehm gestelde uitdagingen de stormachtigste ontwikkeling doorgemaakt en tevens de meest spectaculaire resultaten geboekt en de associatie van Computer (!) Science met electrotechniek vindt dan ook zijn wortels in die eerste periode.

De relatie met de wiskunde is natuurlijk ook heel oud -zeker als we door de logica gemakshalve bij de wiskunde onder te brengen, Turing in de weegschaal leggen- maar "dunner". De numerieke wiskunde speelde in het gebruik een vrij vooraanstaande rol, omdat dit de voornaamste discipline was, die reeds over een arsenaal informatieverwerkende algorithmen beschikte; omgekeerd was de komst van de rekenautomaat, waarbij efficiencynormen zo anders lagen dan bij de tafelmachine, voor de numerieke wiskunde een enorme stimulans om naar nieuwe methoden te zoeken, een stimulans die, als ik goed ben ingelicht, dan ook tot een duidelijke opbleoi van de numerieke wiskunde aanleiding heeft gegeven. Wanneer ik desondanks de relatie met de wiskunde veel "dunner" noem, dan is dat ten eerste omdat de rest van de wiskundige wereld (die nog steeds aarzelde of numerieke wiskunde een legitiem, een onecht of een aangenomen kind van de Regina Scientiarum was) door deze ontwikkeling niet zo beroerd werd, ten tweede omdat behoudens het

motiveren van de behoefte aan snellere en grotere apparatuur de numerieke methoden zelf weinig invloed op machinestructuur gehad hebben.

In de volgende fase, die met de komst van de UNIVAC is ingezet, komt de rekenautomaat als commercieel serieproduct op de markt. Van dit moment af is de ontwikkeling zo stormachtig en zo veelzijdig, dat ik veel vluchtiger te werk moet gaan en elke hoop op volledigheid moet laten varen. Als ooit duidelijk wordt dat "A computer is a many-sided thing." maar al te waar is, is het in dit stadium van de ontwikkeling.

Ten eerste ontwikkelt de computer zich tot een marktproduct, een ontwikkeling waarmee fabelachtige investeringen gemoeid zijn: terwijl deze ontwikkeling sommige fabrikanten opstuwt tot giganten, ervaren vele anderen het als een (te) moeilijk product, te moeilijk om te maken, te moeilijk om te verkopen of beide. Ik noem dit commerciele aspect, omdat het, direct of indirect, het denken van een heleboel mensen vaak ernstiger beinvloed heeft, dan zij lijken te beseffen. De hoge prijs van deze machines heeft in het begin van de zestiger jaren tot een bijzonder eenzijdige opvatting over het idee "efficiency" aanleiding gegeven en krachtige residuen van die eenzijdigheid vindt men nog steeds. De computer is ook een product, dat tot een heel speciale verkoopstechniek geleid heeft en zelfs de verkoopskreten hebben de neiging ons denken ernstig te beinvloeden. (Vele gebruikers en kringen, die een computer moeten uitzoeken, hebben oprecht het idee, dat "compatibiliteit" een heel groot goed is. Historisch gezien is het in elk geval een sales-gimmick en hoeveel mensen nemen de moeite om de potentiele vruchten van de compatibiliteit zorgvuldig af te wegen tegen de onmiskenbare lasten?)

Het is de periode van de opbloei van de administratieve toepassingen, die, hoewel "administreren" in allerlei opzichten een stuk ingewikkelder is dan wetenschappelijke toepassingen, zich in hoofdzaak buiten de universitaire gezichtskring voltrekt; sommigen zullen haasten hier aan toe te voegen "met alle gevolgen van dien". Wanneer Euwe, zelf geen specialist op het gebied van rekenmachines, door een aantal Nederlandse economische hogescholen aangetrokken wordt, om de informatica te doceren zal hij zich beperken tot management aspecten, er van uitgaand, dat zijn studenten het eigenlijke werk aan lagere krachten kunnen overlaten. Van de opmars van de rekenautomaat in de administratieve wereld moet men in retrospectie opmerken, dat hij iets geforceerds gehad heeft: het gebruik heeft jaren lang in het teken gestaan van de vraag hoe men zich zo goed mogelijk met magnetische banden behelpt, ook in die gevallen waar dit duidelijk een ontoereikend medium was. Dan toch maar!

Het is ook de periode geweest -en we zijn tegenwoordig wel eens geneigd dit te vergeten- waarin machines behalve een orde van grootte sneller en groter een orde van grootte betrouwbaarder werden. Die grotere betrouwbaarheid heeft een enorme onvloed gehad, omdat nu veel geraffineerdere methoden van machinegebruik mogelijk werden en het [illegible] aanvankelijke niveau van trivialiteit ontstegen werd. De komst van hogere programmeertalen is hiervan het duidelijkste symptoom. Dit heeft in tweeerlei opzicht de wiskundige bemoeienis met rekenmachines versterkt. Met de komst van ALGOL 60, en speciaal het gebruik van BNF ter definitie van een syntaxis, werd het probleem van compilerbouw in een zodanige vorm gepresenteerd, dat althans gedeelten daarvan zich voor wiskundige behandeling leenden, terwijl dankzij hogere programmeertalen het programma zelf als wiskundig object dat bestudeerd zou kunnen worden, zich aandiende.

Met de komst van hogere programmeertalen hebben zich allerlei kringen van gebruikers ontwikkeld, waarvan vooral "the bubble chamber boys", die dankzij de prijs van hun overige apparatuur met veel volume konden schreeuwen, zich aan allerlei universiteiten tot krachtige pressiegroepen ontwikkeld. Het feit, dat zij zo ongans veel rekentijd konden verstoken, wettigt natuurlijk de vraag of hun experimentele techniek niet voor enige herziening in aanmerking komt, maar deze vraag is niet erg populair.

Volledigheidshalve noem ik de toepassing van rekenmachines in instrumentatie en overige besturingsproblemen. Het is wederom de grotere betrouwbaarheid van de hardware, die dit mogelijk maakt. Op het denken van de computer scientist heeft deze ontwikkeling niet zo erg veel invloed gehad. Dit is gedeeltelijk te wijten aan het feit, dat het hier toepassingen van vaak kleinere machines betrof. De belangrijkste oorzaak is ongetwijfeld, dat mensen met "real time toepassingen" meer dan de meeste andere machinegebruikers aan het gebruik van machinecode hebben vastgehouden. Je hoort vaak de stringentheid van de real time verplichtingen als verklaring en rechtvaardiging van deze terughoudendheid om van hogere programmeertalen gebruik te maken. Ik geloof, dat het fairder is om de afwezigheid van gezonde synchronisatieprimitiva in allerlei hogere programmeertalen gezien moet worden als symptoom van relatief onbegrip van de hele synchronisatieopgave en dat het dit onbegrip is, dat in iedere speciale opgave maakt, dan men zo vaak weer naar een ad hoc oplossing zoekt.

Naast de bloei van de programmeertalen wordt het afgelopen decennium door twee ontwikkelingen gekarakteriseerd, LSI (Large Scale Integration) en de moeilijkheden bij de bouw van operating systems. LSI is in eerste instantie een fabricagetechniek voor circuitry, waarvan de belofte is, dat rekeneenheden vele malen goedkoper gemaakt zullen kunnen worden, dan nu het geval is. LSI is nu al vijf jaar bezig, deze belofte in te lossen en het gaat niet zo hard, als ons destijds is voorgespiegeld. Hiervoor zij allerlei technische problemen verantwoordelijk (precisie bij de fabricagetechniek, warmteontwikkeling etc.); qualiteitscontrole wordt, alsof het al niet genoeg is, ook een levensgroot probleem. Als deze technische problemen eenmaal opgelost zijn, blijft er nog een aanzienlijk probleem over: hoe concipieren we een rekenproces met "distributed activity"? Het is weer een duidelijk gebied, waar concurrentie een ontwikkeling lijkt te forceren. De ontwikkeling van operating systems is ontstellend tegengevallen en dit heeft tot ten minste drie "lines of thought" aanleiding gegeven. Ten eerste, dat de bouw van iets ingewikkeld als een operating system een enorme intellectuele uitdaging inhoudt, en dat je inzicht in de veelzijdigheid van de probleemstelling voor dat je aan het operating system begint geleid moet hebben tot een grote mate van conceptuele klaarheid, zonder welke het project gedoemd is in de chaos te verzinken. Ten tweede, de opvatting dat operating systems qualitate qua zo ingewikkeld en in hun gedrag zo onvoorstelbaar moeten zijn, dat we hier "experimentele computer science" moeten bedrijven, dwz. van het gebeuren een model moeten maken en vervolgens, als in elke andere natuurwetenschap, door meting dit model moeten verifieren. Ten derde, dat, omdat we nu weten dat een echt moeilijk programma alleen maar gemaakt kan worden door een leger van een paar duizend programmeurs, de hoofdopgave van programmeren een kwestie van organisatie techniek is.

Ten leste moet ik een veranderende instelling tegenover programmafouten vermelden. Vijf jaar geleden was het niet ongebruikelijk om in vertrekken, waar geprogrammeerd werd, een klein wandbordje aan te treffen met de tekst (onder een sprekend plaatje) "This is a program bug; be kind to the little animal, it gives you job security." Voor de relatie tussen software house en "gevangen klant" was dit inderdaad het geval: het software house maakte een product, dat goed genoeg was, opdat de klant er mee ging werken en afhankelijk van werd, anderzijds was het product van een zodanige kwaliteit, dat het product

constant onderhoud vergde en dit onderhoud alleen door de oorspronkelijke leverancier gedaan kon worden. Maar het tij is aan het keren. Een van de oorzaken is ongetwijfeld het geaccumuleerde ongenoegen van aldus gevangen klanten. Een tweede oorzaak is te vinden in de weliswaar schuchtere, maar met veel succes bekroonde pogingen om correctheid van programma's te bewijzen. Een derde oorzaak is een professioneel gevoel van "uneasiness", nl. de overweging dat wanneer (en dit gebeurt!) systemen een steeds belangrijkere plaats in onze samenleving gaan innemen, er binnen afzienbare tijd natuurlijk eens een keer een ramp gebeurt (en er zijn mensen, die zeggen, dat dat al gebeurd is!), waarvan de oorzaak onder andere herleid kan worden tot een profane programmafout. Deze vrees ligt ten grondslag aan de bezorgdheid van bv. de BCS om te komen tot "professional rules of conduct"; bij een recent congres in Amerika zijn er stemmen opgegaan om de Amerikaanse overheid te bewegen geen software producten zonder correctheidsbewijs te accepteren. Hoopvol als deze geluiden enerzijds zijn, ze zijn ook beangstigend: het zou voor de zoveelste keer de aanleiding voor een geforceerde ontwikkeling kunnen zijn (compleet met alle vormen van "overselling".)

Nou, dit is, waar we zowat staan. "A computer is a many-sided thing" en afhankelijk van de kant, waartegen we aankijken, kunnen we computer science beschouwen als

a) een verlengstuk van de electrotechniek
b) een verlengstuk van de numerieke wiskunde
c) een verlengstuk van bedrijfskunde
d) als een verlengstuk van instrumentatie
e) als een verlengstuk van systems theory
f) als een verlengstuk van de wiskunde.
g) etc.

Het is duidelijk, dat de enige manier om met dit soort bespiegelingen iets te kunnen aanvangen het nader analyseren van de vage kreet "verlengstuk" zal zijn. Ik wil daarom proberen de aard van de interface tussen de informatica enerzijds en aangrenzende gebieden anderzijds iets nader te beschrijven. Het idee, dat hier achter ligt is dat de levensvatbaarheid van een "isoleerbare" discipline sterk beinvloed wordt door de mate van isoleerbaarheid, en, indien de isoleerbaarheid mogelijk is, ook doorde noodzaak tot isolering, een noodzaak die zou kunnen ontstaan als de geestelijke disciplines aan beide zijden van de interface heel verschillend zijn. Andere criteria ten aanzien van levensvatbaarheid, die ik bedenken kan, hebben te maken met omvang: een in principe isoleerbaar vak kan "te klein", in die zin, dat het heel gewoon is, dat iemand uit een aanverwant gebied "het er wel even bij doet", het kan qua werkterrein te arm zijn. Hoe we al deze dingen precies zouden moeten wegen, weet natuurlijk geen kip, maar dat ontheft mij voor mijn gevoel niet van de plicht te proberen hier enig gevoel voor te ontwikkelen en te beschrijven.

Om te beginnen de interface tussen informatica en electrotechniek. Ik zie hier drie, voor mijn gevoel vrij duidelijk gescheiden aspecten. Ten eerste de historisch oudste, nl. als leverancier van de technologie, die de fabricage van computers mogelijk gemaakt heeft, een fabricage waardoor de problematiek van de informatica actueel is geworden. Deze interface, waarin de electrotechniek zich in de informatica doet gelden via de existentie van machines, [illegible] is het medium voor een wederzijds zeer zwakke koppeling: in het overgrote gedeelte van de informatica is het volslagen irrelevant dat machines via hoofdzakelijk electronische technieken gerealiseerd worden, in dat gedeelte van de electrotechniek dat zich bezig houdt met de constructie van circuitry doet het er wederom heel weinig toe, waarvoor de te maken hardware uiteindelijk ingezet wordt.

Het tweede aspect van de interface tussen de electrotechniek en de informatica wordt gegeven door de omstandigheid dat ontwerp van een niet-triviaal stuk hardware, annex mogelijke automatisering van hetzij stadia van het productieproces, hetzij bewaking van correctheid, up to date houden van documentatie etc. een enerzijds zo complexe zaak is, terwijl men anderzijds het gevoeld heeft, dat veel van de (vaak discrete) informatie zich wel voor automatiesche verwerkingsprocedures zou lenen. Hierdoor wordt "het electronisch ontwerpen" mogelijkerwijs een vruchtbaar werkterrein voor de informaticus. We signaleren dit aspect van de interface, maar laten voorlopig de educatieve consequenties hiervan in het midden: men kan proberen in deze toepassingsmogelijkheid van de informatica te voorzien door "informatica" een basisvak (een basisbijvak) van de toekomstige electronisch ontwerper te laten zijn, dan wel stukken electrotechniek en leer van de automatisering als potentieel bijvak op te voeren voor de afstuderende informaticus, die aan ziet komen, dat hij zich op dit toepassingsgebied wil gaan inzetten.

Het derde aspect van de interface tussen de electrotechniek en de informatica wordt geleverd door de opgave van "functionele specificatie van een nieuw hardware ontwerp". Zodra we veronderstellen, dat de functionele specificatie niet door commerciele overwegingen volledig gedicteerd worden, maar technische overwegingen ook een rol mogen spelen, dan krijg je bij deze specificatietaak heel duidelijk twee aspecten: de vraag van de informaticus "kan ik zo'n machine gebruiken?" en de vraag van de electrotechnicus "kan ik zo'n machine maken". Weer is, door de bank genomen, de interactie zwak, in die zin, dat het grootste gedeelte van de informatici (althans op het ogenblik) niet met deze vraag tot functionele specificatie van een stuk hardware worden uitgenodigd (sterker: in een heleboel organisaties wordt dit denken zelfs helemaal niet aangemoedigd!), terwijl bij de ontwerper van circuitry de aard van zijn werk weinig door de functionele specificatie wordt beinvloed. Een en ander neemt natuurlijk niet weg dat voor een goed verdedigbaar ontwerp een goed samenspel tussen beide disciplines vereist is.

Wat heeft dit alles te maken met onderlinge isoleerbaarheid? Het eerste aspect, als dat het enige was, maakt een perfecte isoleerbaarheid mogelijk, de informaticus werkt in de wereld waarin de idealisering van de discretisering voltooid is, de electrotechnicus leeft in de wereld waarin met in zijn ogen wezenlijk analoge middelen deze discretisatie gesimuleerd moet worden. De andere twee aspecten van de interface hebben duidelijk een interdisciplinair karakter. Bij de vraag of we een dergelijk werkterrein liever bij (beoefenaren) van de ene, dan wel de andere discipline willen onderbrengen, moet je de vraag beantwoorden, welke aspecten van de asymmetrie je in het geding wilt brengen. Ik dacht, dat "relatieve beheersing van het eigen vak" hierin een belangrijke rol moest spelen in die zin, dat de grensactiviteit hoofdzakelijk ligge op het gebied van de man, die relatief zijn vak het slechtste beheerst: de ander kan zijn kant van het contract nl. zo duidelijk formuleren. In het geval van design automation ten bate van het electronisch ontwerpen zou ik me voor kunnen stellen, dat het soort informatietechnische problemen op betrekkelijk standaard [illegible] manieren kan worden opgelost, terwijl de vraag "hoever komen we met onze automatisering?" a priori veel onduidelijker is. Van een project voor design automation voor het electronisch ontwerpen verwacht ik eerder baanbrekend werk van de electrotechnicus, dan van de informaticus. Mijn neiging zou daarom zijn, deze activiteit voornamelijk te zien als een electrotechnische aangelegenheid? waarin, als informatici er aan meewerken, de informatici niet de leidende rol zouden mogen hebben. Bij de taak van de functionele specificatie van een hardware machine zou het wel eens precies andersom kunnen liggen: de hardware man, die inmiddels zijn beperkingen zo goed kent, zou een gegeven ontwerp wel eens veel sneller op maakbaarheid kunnen taxeren, dan dat de software man het ontwerp op bruikbaarheid taxeren kan. In mijn neiging het hoofdaccent van de ontwerpverantwoordelijk-

heid zo mogelijk te leggen bij de relatief meest incompetente discipline (daar kan je je immers de hardste builen vallen!), zou ik bij het ontwerp van de functionele specificatie bij mijn huidige taxering van de relatieve competenties de hoofdverantwoordelijkheid bij de informatica-kant willen leggen. Technieken ter verantwoorde evaluatie van de bruikbaarheid van voorgestelde hardware configuraties is duidelijk een onderontwikkeld gebied van de informatica. Pas als de informatica die heeft en aan de electrotechnicus kunnen doceren, is machineontwerp van uit de kant van de informatica een delegeerbare activiteit.

Wij vervolgen onze beschouwingen en komen nu toe aan de interface tussen de informatica en de numerieke wiskunde. Na aanvankelijk als practisch enige leverancier van algorithmen gefungeerd te hebben, is in de latere jaren de rol van de numericus wat op de achtergrond geraakt. Een belangrijke bijdrage heeft de numerieke wiskunde daarna nog gehad door met redenen omkleed fatsoenlijkheidsnormen voor drijvende arithmetiek op te stellen. (Dat de numerici hierbij vaak roependen in de woestijn waren, kunnen we betreuren, ik beschouw hun bijdrage in dezen desalnietemin als "belangrijk" omdat ze, zij het op een gedetailleerd punt, een qualiteitscriterium hebben aangedragen, waaraan een machineontwerp getoetst kan worden.) Als machines, waarin een hoge mate van parallelle rekenactiviteit plaats kan vinden, reeel worden, zou gedurende een aantal jaren de interferentie tussen machinebouw en -gebruik enerzijds, en de numerieke analyse anderzijds heviger kunnen worden. Het is niet ondenkbaar dat weer de numerieke wiskunde de discipline is, die een tijd lang als voornaamste leverancier van (nu parallelle) algorithmen op zal treden; en het is evenzeer denkbaar, dat de numerieke wiskunde van zijn kant hier een grote stimulans zal ondervinden. Ik kan het niet goed bekijken: als we de ILIAC IV als mogelijk symptoom van deze ontwikkeling beschouwen, dan wijst dat stuk recente geschiedenis niet sterk in die richting, maar je kan nooit weten. Duidelijk is, dat er allerlei machinetoepassingen zijn, waarbij een goede kennis van de bestaande numerieke wiskunde een vereiste is. Wanneer ik de numerieke wiskunde niet als een "must" voor de informaticus taxeer, dan is dat ingegeven door de volgende overwegingen. Ten eerste heeft zich er een ruim en gevarieerd toepassingsgebied van rekenautomaten ontwikkeld, waar de simulatie van het continuum helemaal geen rol speelt, anderzijds is "the body of knowledge", die zich onder de naam numerieke wiskunde heeft ontwikkeld, inmiddels zo omvangrijk (en hecht technisch verknoopt met een paar andere takken van de wiskunde), dat het absoluut irreeel is, om van elke informaticus te eisen, dat hij een all round numericus en passant er bij is. Ik neem op quantitatieve gronden aan, dat we ons er bij neer moeten leggen, als de doorsnee informaticus enig benul van de numerieke wiskunde heeft (genoeg om te weten, wanneer hij een numericus in de arm moet nemen), terwijl je omgekeerd hoopt dat de man, die echt moeilijk numeriek werk doet, voldoende benul van de informatica heeft, om doorgaans verantwoord machines te gebruiken (en, op zijn beurt, als de situatie zich voordoet, niet schroomt om een informaticus in de arm te nemen, als de organisatorische moeilijkheden hem bijvoorbeeld uit de hand lijken te lopen).

De interface tussen informatica en bedrijfskunde is een stuk ingewikkelder. Ik kan zeker vier raakvlakken bedenken. Het eerste raakvlak is betrekkelijk economisch en is het raakvlak met de computerindustrie, die de computer (hopelijk met winst) als product op de markt wil brengen; als men aan dit aspect veel gewicht toe wil kennen, zou men tot de stellingname kunnen komen, dat de "realistische" informaticus zich dan wel niet hoeft te beperken tot de bestaande apparatuur, maar bijvoorbeeld wel tot de "verkoopbare"... De drie andere raakvlakken betreffen meer de organisatieleer. Ten eerste fungeert elk gemaakt systeem als onderdeel van een grotere organisatie. Voor de wetenschappelijke gebruiker, die een programma voor zichzelf maakt omdat hij een differentiaalvergelijking opgelost zou willen hebben, is die "omvattende organisatie"

niet zo groot en niet zo onoverzichtelijk; voor hem althans. Het aantal dusdanige gebruikers is ontzettend groot en het is een ØØØ gevarieerd stel, zo gevarieerd, dat je nauwelijks van de centrale informatica kunt verwachten, dat die een overzicht heeft van hoe het gebruik van automatische rekenapparatuur de wijze van werken in al die verschillende disciplines zal beinvloeden. Je kunt daarom alleen maar hopen, dat die gevarieerde gebruikers zelf, ieder voor zich, er langzamerhand achter komen, wat het fenomeen "computer" mogelijk voor hun eigen vakbeoefening te betekenen heeft. Zodra een computer een essentieel onderdeel gaat worden in een grotere organisatie -en dit mag een universitair rekencentrum zijn, dat "gerund" moet worden, het mag een stuk automatische informatieverwerking in een ziekenhuis zijn (voor mijn part inclusief ØØØØØØØ patiëntenbewaking en bewaking van de apotheek!) of voorraadbeheer en planning van een fabriek- dan wordt de interactie tussen systeem en omvattende organisatie ontegenzeggelijk veel inniger. Ten tweede ØØØØØ vertoond elk groot programma -en een operating system is hiervan misschien wel het duidelijkste voorbeeld- zelf de structuur van een organisatie (en wordt in overeenstemming daarmee dan ook vaak beschreven en begrepen in een terminologie, die aan de organisatieleer ØØØØØØØ ontleend is). Ten derde is het maken van een groot programma een bezigheid, die zich in elk geval over een flinke periode uitstrekt, gedurende welke periode er heel vaak een aantal mensen bij betrokken zijn: kortom, het maken van programma's heeft zelf duidelijk organisatorische aspecten. Gelukkig ligt in deze laatste twee gevallen de wijze van interactie tussen informatica en organisatieleer wat duidelijker, ik heb nl. sterk de indruk, dat er in dezen slechts sprake is, resp. hoort te zijn, van één-richtingverkeer. Dat een groot programma zelf een organisatie is, is voor de programmeur vanzelfsprekend, maar hij heeft daarbij weinig van de gevestigde organisatieleer te leren, die zich zelden of nooit kan veroorloven allerlei sociale randcondities niet in de beschouwing te betrekken en het resultaat lijkt te zijn, dat het omgekeerd voor de organisatiedeskundige interessanter is, om eens een groot operating system te bestuderen, dat hem confronteert met een complexe organisatie "in reincultuur". Wat de organisatie van de programmeerarbeid betreft moeten we opmerken, dat de software afdelingen, die zijn opgezet volgens de geldende regels en normen van de organisatieleer, om het voorzichtig uit te drukken, niet zo erg succesvol zijn geweest: het klakkeloos uit andere productieprocessen overnemen van een aparte groep die verantwoordelijk is voor "quality control", bv. heeft tot absurde en onmogelijke taakverdeling geleid. Ik geloof, dat we veilig stellen kunnen, dat het abstracte product "software" met al zijn interne logische verknopingen een heel ander product is dan wasmachinemotoren en dat de manier, waarop men het constructieproces van software moet organiseren meer bepaald dient te worden door de specifieke eigenschappen van software in het algemeen en het onderhavige product in het bijzonder, dan op zo anderssoortige omstandigheden gefundeerde standaard organisatorische opvattingen. Voor de software engineer beperkt de interactie zich in eerste instantie tot het feit, dat hij leren moet, dat hij zich in zijn latere leven geen inadequaat organisatiepatroon moet laten opdringen, de organisatiedeskundige zou er heel veel van kunnen leren, als hij er zich eens zorgvuldig in verdiepte, hoe een succesvol team een software project klaart; misschien dat hij dan het wezenlijke verschil tussen abstractheid en vaagheid leert apprecieren. Het samenspel tussen "systeem" en omvattende organisatie blijft een teer punt. Ik zou me kunnen voorstellen, dat men de vraag "Hoe beinvloede automatiseringsmogelijkheden een organisatie?" in zijn algemeenheid aan organisatiedeskundigen (wie en wat dat ook moge zijn) moet over laten, terwijl de omgekeerde vraag "Hoe kies, resp. dimensioneer ik een installatie, gegeven de omstandigheden, waaronder hij zal moeten werken?" -een vraag, waarmee bv. de directie van ons rekencentrum heel duidelijk mee geconfronteerd wordt- wel tot het terrein van de Informatica rekent.

Toen ik de instrumentatie opschreef, was ik in eerste instantie getemp-teerd om "de experimentele natuurkunde" op te schrijven, er van onder de indruk hoe in sommige universiteiten het rekenbedrijf geterroriseerd wordt door "the bubble chamber boys", die zich als grootste klant de grootste zeggenschap in keuze aanmatigen, een keuze die louter en alleen bepaald wordt door het verlan-gen compatible te zijn met hun Amerikaanse vriendjes. Iedereen begrijpt, waar dat op uitmondt. Ik zou tegenover die physici willen opmerken, dat het feit dat zij zo ongans veel machinetijd verstoken -en voor het claimen van die tijd chanteren ze de maatschappij met de hoge prijs van hun overige apparatuur!- wel eens een indicatie zou kunnen zijn dat er aan hun experimenteertechniek iets schort. Wat betreft de overige "instrumentatie", ik denk dan aan allerlei min-of-meer real time toepassingen, zoals het bewaken van processen en patienten, het verzamelen van stromen meetgegevens in een experimentele opstelling en wat onder de verzamelnaam "message switching" bekend staat (inclusief netwerken). Het lijkt mij buiten kijf, dat dit gebied tot de informatica gerekend moet wor-den: in alle gevallen, die ik hier bedacht heb, zijn de uitwendig opgelegde specificaties vrij goed formuleerbaar (medische kennis is echt niet nodig); en dat, omdat je nu veel sneller reageren kunt, andere indicaties medisch belangrijk kunnen worden, wel, dat is iets, dat je een medicus nu nog wel kunt uitleggen.

Systems theory heb ik er maar bij geschreven om der wille van de volle-digheid. Ik moet hier bekennen, dat ik er niet zo heel veel van af weet. Ik sta er dan ook met gemngde gevoelens tegenover. Ik heb, ruwweg drie ervaringen. De eerste is al weer een hele tijd geleden: toen hield systems theory zich nadrukkelijk bezig met systemen, waarvan het gedrag door lineaire differentieel-vergelijkingen (door wat anders!) beschreven werd; afgezien van enige stabili-teitsbespiegelingen heb ik er toen niet veel relevant uit kunnen halen. Het tweede contact was hoopgevender: dat was toen ik het verhaal van de "allowance counts" aan Hautus kon slijten en wat ik over het bijhouden van de geschiedenis bedacht had, in de besturingstheorie een gevestigde methode bleek te zijn. Dat is dan een ogenblik, waarin je een mogelijke kruisverbinding proeft. Mijn laat-ste contact was met een boek van drie Amerikanen van Oost-Europese origine (zoiets als "Mesarovic" et.al.) van Case Western Reserve University, waar sinds tien jaar een instituut voor systems study bestaat. Het boek gaat over hierarchische systemen, het meest opvallende is, dat in hun inleiding, waarin ze drie begrippen "hierarchie" willen uitleggen, deze drie begrippen niet erg duidelijk over het voetlicht komen. De rest van het boek lijkt een moeizaam wiskundig apparaat, waar weer allerlei lineariteitsveronderstellingen om de hoek komen kijken. Ik heb sterk de indruk, dat voordat dit relevant is, er nog wel een paar dingen moeten gebeuren: ten eerste mag er wiskundig nog wel wat verfraaid worden, ten tweede mogen er nog wel een paar Koestlers komen, om de quintessence er uit te distilleren.

De relatie met de rest van de wiskunde is natuurlijk iets, waarover we, uit hoofde van de positie, waarin we ons bevinden, iets explicieter kunnen en moeten zijn. (Omdat ik de numerieke wiskunde al heb aangestipt, laat ik die nu verder buiten beschouwing.)

Om te beginnen is er een heel duidelijke verbinding met wat hetzij "logica", hetij grondslagenonderzoek genoemd wordt. Je denkt hier in de eerste plaats aan Turing, Post etc. ; enige bekendheid hiermee hoort m.i. ongetwijfeld tot gewenste geestelijke bagage van de informaticus: het is een respectabel stuk wiskunde en komt, zij het niet frequent, af en toe heel significant om de hoek kijken. (Ik denk aan het artikel van Floyd, waarin hij van een algemene klasse BNF-grammatica's aantoont, dat de aan-of afwezigheid van de mogelijkheid tot syntactische dubbelzinnigheden niet decideerbaar is. Sindsdien is meer naar voren gekomen, dat de grammatica van een programmeertaal uit het oogpunt van

"human engineering" nog wel aan heel wat stringentere eisen moet voldoen, waardoor de studie van dergelijke "pathologische" grammatica's wel wat aan betekenis heeft ingeboet. Het zij zo,hier ligt desalnietemin voor mij een relevant stuk theorie.) De terugkoppeling vindt ook andersom plaats, nl. zodar wiskundigen zich gaan afvragen, welk gedeelte van hun "symbol manipulation" zij aan een automaat kunnen delegeren en met welke pretentie. Hier kom je direct tegen het grondslagenonderzoek aan, resp. de tak van dienst, die elders "artificial intelligence" genoemd wordt. Automath is duidelijk een symptoom van deze interactie. Het komt mij voor, dat in dit soort projecten de informatica hoofdzakelijk "toeleverende partij" is totdat het project heel groots wordt: dan kan het het karakter van "a highly unusual application" krijgen, met alle repercussies van dien. Het is het vermelden waard, dat dit soort machinegebonden projecten sinds de inventie van LISP niet erg veel nieuwe invloed op de informatica gehad heeft!

Het afgelopen decennium heeft sterk in het teken van de hogere programmeertaal gestaan. Men is begonnen met allerlei syntactische verkenningen en het daarvoor benodigde wiskundige apparaat (en naar mijn smaak: meer dan dat) is ontwikkeld. De volgende stap is geweest de poging om semantische definitie van programmeertalen te formaliseren; het werk van de Weense groep mag hier niet onvermeld blijven. Terwijl dit werk in principe als heel belangrijk getaxeerd moet worden, kunnen we alleen maar betreuren, dat deze ontwikkeling aan de formalisering van PL/I is opgehangen. Een andere tak van activiteit is ontsproten aan het werk van Dana Scott. Dit lijkt me conceptueel veel eleganter, mijn kennis van lattices is beslist onvoldoende om het allemaal naar waarde te kunnen schatten. Bij wat ze dan uiteindelijk bereikt hebben, heb ik een licht gevoel van teleurstelling helaas niet kunnen onderdrukken.

De bezorgdheid om "correctheid" treft mij als de jongste ontwikkeling. Er zijn hier twee hoofdstromingen: de ene betreft de ontwikkeling van formele bewijstechnieken, de andere betreft de programmastructurering, zodat de bewijsvoering vergemakkelijkt wordt. Hoewel ieder, die zich met dit soort werk bezighoudt, wel duidelijk in een van beide kampen pleegt te zitten, lijkt het me duidelijk, dat deze twee ontwikkelingen hand in hand horen te gaan. (En ook doen.)

Ik verwacht -maar dat heb ik al vaker gezegd- de belangrijkste interactie tussen de informatica en de rest van de wiskunde meer op het "quo modo" dan het "quod". Als ik zie, hoeveel moeite ik met allerlei wiskundige teksten kan hebben, louter en alleen omdat ze zo slordig geschreven zijn en de spelregels, volgens welke zo gelezen moeten worden, zo vaag zijn -deze moeilijkheid mijnerzijds is natuurlijk een "déformation de métier"-, dan denk ik vaak, "ik wou, dat de auteur iets intensiever met de informatica geconfronteerd geweest was"; omgekeerd kan ik allerlei programmeursgeschriften tegenkomen, waarvan ik dan denk "ik wou, dat de man zich een beetje meer als wiskundige tegenover zijn problematiek opstelde".

Ik wou het overzicht van het contact met de randgebieden hier eigenlijk maar bij laten en proberen of ik, geinspireerd door deze verkenningen, het voor de informatica specifieke werkterrein, de specifieke geestelijke discipline en de wetenschappelijke pretentie isoleren kan. Nogmaals: de vraag in hoeverre we op grond van de locale omstandigheden in een curriculum recht kunnen doen wedervaren aan dit beeld van de informatica, is een vraag, die ik slechts in tweede instantie -en niet in dit stuk!- wil bekijken.

Laat mij nu trachten het typisch eigene van de informatica te vinden. Ik wil hier vooropstellen dat ik denk aan de informatica in de omgeving van een technische hogeschool. In universitaire omgeving zou men nl. een minder constructieve opvatting van informatica kunnen huldigen en bv. het soort informaticus willen opleiden die vervolgens zijn sporen verdient in een team van medici etc., dat een studie maakt van de informatieverwerking door het menselijk oog en de daarachtergelegen hersenen. Hoe legitiem dergelijk onderzoek ook kan zijn, ik wilde de taakstelling van de informatica in het raam van een technische hogeschool niet daarnaar uitstrekken.

Dab focussert de informatica zich voor mij op het beheerst ontwerpen van complexe mechanismen die in voorspelbare mate beantwoorden aan een zorgvuldig gekozen doel. En als je nou zegt "maar er worden toch veel meer "systemen", "mechanismen" ontworpen, die aan een of ander doel moeten beantwoorden? Hoe kan je hier iets specifieks voor de informatica uit destilleren?", dan zou ik het volgende willen antwoorden.

Het veld van de informatica is in dit opzicht iets aparts, omdat ik geen ander gebied weet, waarin de te ontwerpen mechanismen enerzijds zo ongelofelijk ingewikkeld kunnen zijn -dit is een onmiddellijk gevolg van de macht van tegenwoordige rekenapparatuur- terwijl anderzijds de praemissen, op grond waarvan het mechanisme zou moeten werken, zo simpel en expliciet bekend zijn, zodat volledige intellectuele beheersing, die niet alleen nodig is, ook mogelijk is. Het is de enorme spanwijdte tussen de eenvoud van de basis en de sophistication van het uiteindelijke product, dat voor mij het unieke kenmerk van de informatica is. Het is een onmiddellijk gevolg van de aanwezigheid van machinerie, die zo ongeveer een millioen stuurbare instructies per seconde uitvoert. En het is daarom, dat ik de taakstelling van de informatica zie als een intellectuele uitdaging zonder precedent in de geschiedenis van onze cultuur.

Hier komt nog een heel bijzonder facet bij: het is een werkterrein waarin, meer naar het mij voorkomt dan in allerlei andere disciplines, elke routineactiviteit "op de schopstoel zit": zodra in de uitoefening van de informatica iets een duidelijke routineactiviteit begint te worden, bestaat er een goede kans, dat het geautomatiseerd wordt. Dit hebben we in het verleden al een paar keer gezien -de "codeur" bestaat eigenlijk nauwelijks meer, het is in elk geval geen baan met houvast- en de aard van het vak maakt, dat we dit in de toekomst ook nog wel eens mogen verwachten. De onbestendigheid van routinetaken is een van de facetten, die mij huiverig maakt om mensen van onvoldoende niveau in de informatica te trekken: de uitvlucht is dan altijd "je hebt mensen van allerlei niveau nodig", maar in de informatica kon wel eens een vacuum ontstaan tussen de echte top en de echte onderlaag (van ik zou bijna zeggen "operateursniveau").

Terug naar de intellectuele uitdaging, dat is een prettig houvast (vind ik tenminste). Het betekent, dat alle onmisbare kennis en vaardigheid om tegen deze uitdaging zo goed mogelijk opgewassen te zijn, een [illegible] wezenlijk onderdeel van de informatica uitmaakt. Wat zijn die vaardigheden, wat is die kennis? Hier bemerk ik, dat ik me in een merkwaardige hoek gemanoeuvreerd heb in die zin, dat ik nu nauwelijks meer met kennis kan komen aandragen, waar ik zelf niet redelijk in thuis ben, noch met vaardigheden, die ik niet redelijk beheers. Staat een beetje gek!

Kennis van machines is essentieel, althans van hun functionele specificatie. Zo als ik me voortreffelijk een informaticus kan voorstellen, die niets van de electronica afweet, kan ik als ik een beetje mijn best doe, me ook wel een informaticus voorstellen, die eigenlijk nog nooit een machinecode gezien heeft. Ik zou dit niet toejuichen, maar voorstellen kan ik het me wel.

Ik zou het daarom niet toejuichen, omdat ik over bestaande machines nog niet zo enthousiast ben en de informaticus gaarne als gesprekspartner zag op zoek naar betere machines. Ik kan het vervolgens niet toejuichen, omdat de bestaande alternatieven (abstracte machines in de vorm van een programmeertaal) nu ook niet zo ideaal zijn en ik de informaticus desgewenst in staat zou willen achten een betere taal te ontwerpen en te implementeren. Functionele specificaties van bestaande machines -althans in grote lijnen- behoort kennelijk tot zijn geestelijke bagage. (Nou, dat is fijn, want we zullen moeite genoeg hebben om in de informatica genoeg relevante, doceerbare "kennis" onder te brengen.) Je kunt hier weer twee kanten uitgaan: ik kan me de informaticus voorstellen, die een groot aantal machines kent, liever heb ik de informaticus, die de gangbare aspecten van machines -met hun motivering, hun voors en hun tegens- kent.

Kennelijk is een zekere kennis van de logica en vertrouwdheid met bewijstechnieken een hoeksteen. Ook hier moet relevante, doceerbare kennis te vinden zijn.

Vertrouwdheid met automatiseringstechnieken -hier valt zeker onder het maken van vertalers, assemblers en dergelijk soort spul- is dacht ik ook een vereiste: de informaticus moet zijn eigen "tools of his trade" kunnen maken. Bootstrapping, hetzij op een machine, hetzij van een machine naar een ander, is ook zo'n artikel. Ik wil hiermee niet gezegd hebben, dat je nu zonder meer als informaticus specialist op het gebied van taalimplementatie moet zijn. Ik heb in Maryland een serie voordrachten van David Gries over compiler constructie bijgewoond, maar wat mij daarvan in hoofdzaak is bijgebleven is, dat de complicerende oorzaken heel vaak gezocht moeten worden in, hetzij een ongelukkig gekozen taal (en dat geldt dan niet alleen voor FORTRAN, we weten inmiddels dat ALGOL 60 een iets andere syntax zou kunnen hebben, waardoor het parsing problem een stuk makkelijker zou zijn geweest), hetzij een ingelukkige object machine. Wat dacht ik tot de gewenste bagage van een informaticus behoort is enige vertrouwdheid met de structuur van vertalers en daarbij -zonder dat is de kennis me niets waard- een inzicht in hoe de vertalertaak van de structuur van source en object language afhangt, opdat hij, als hij de keuze heeft, de vertalertaak minimaliseert. (Software is a necessary evil!) Je kunt ook je op het standpunt stellen, dat je studenten zoveel van vertalen van gangbare talen voor gangbare machines moeten weten, dat ze pasklaar in een industrieel vertalerteam kunnen worden ingeschakeld: dat beschouw ik dan als hoger beroepsonderwijs, die kennis kan ik niet tot de kern van de informatica rekenen. In de bovengeschetste critische (bah, modewoord) zin wel, want hij moet desgewenst zijn eigen gereedschap kunnen maken. Op het hier geschetste gebied is ongetwijfeld relevante en doceerbare kennis te selecteren, ik denk hierbij aan Niklaus Wirth die dit aspect van de informatica voor mijn gevoel als geen ander beheerst, speciaal wat betreft de implementatiekant. Op deze kennis kan voor een gedeelte het vermogen berusten om een hardware ontwerp op zijn bruikbaarheid te toetsen, om een voorgestelde taal op zijn implementeerbaarheid te toetsen. (Maar hier heb ik het al over een vermogen, niet meer over kennis.)

Een volgend man, die ik nu onmiddellijk noemen moet is Hoare. Wat wij aan hem danken is de signalering dat de intellectuele beheersbaarheid van programma's (en hun implementeerbaarheid: Hoare claimt, dat de twee vaak hand in hand gaan) critisch kan afhangen van bepaalde linguistische constructies, bv. de preciese structuur van het parametermechanisme. Als je een taal ontwerpt, moet de taal je de combinatorische vrijheden geven die je denkt nodig te hebben: als het jasje te wijd is,en je ook een heleboel andere dingen kunt, kan je in het schip van de onbeheersbaarheid terechtkomen. De alertheid, dat enerzijds elke informaticus de hem gestelde taak moet generaliseren, maar dat hij anderzijds

angstig bewaken moet, dat zijn generalisatie intellectueel hanteerbaar blijft, lijkt me een vitaal onderdeel van de informatica. Ook hieruit is wel relevante en doceerbare stof te destilleren, maar hoe je dat moet doen is niet zo een twee drie duidelijk. Dit kon wel eens een gebied zijn, waar je een hoop bereiken kunt met "negatieve voorbeelden", dwz. de confrontatie met een aantal pitfalls, gewoon laten zien, hoe je je in de nesten kunt werken. (Ik doe dit bij het begin van het college Operating Systems, waar ik sequentiele processen met gebrekkige methoden ogenschijnlijk goed laat samenwerken. Dat werkt didactisch heel goed, de kennis van een aantal van dergelijke pitfalls is waarschijnlijk ook wel een essentieel onderdeel van de informatica.)

Het college Operating Systems, zoals dit bezig is in te klinken, lijkt me ook onmisbare kennis, in die zin, dat ik tegenwoordig me geen competent informaticus kan voorstellen, die van zijn leven alleen maar sequentiele processen heeft gezien op hun eentje, en nooit de samenwerking tussen een aantal van hen. De dan optredende problematiek is moeilijk en onverwacht genoeg om niet te kunnen zeggen "als iemand dat tegen het lijf loopt, bedenkt hij het wel even". Een tweede aspect is, dat door een overzicht te geven van een simpel operating system, de student enerzijds een idee heeft van de structuur van het gereedschap, dat hij aan het gebruiken is en anderzijds met wat nieuwe aspecten geconfronteerd wordt, waaraan hij de adequaatheid van een hardware ontwerp kan proberen te toetsen. Een encyclopaedisch overzicht van een of meer gangbare operating systems voor een of meer gangbare machines lijkt mij wederom volledig misplaatst in een academisch curriculum. Hier moet dezelfde bescheidenheid betracht worden als bij de compilers. (Kennis van de structuur van operating systems is voor een gebruiker van een operating system misschien nog wel essentieler als de kennis van een compiler voor de gebruiker van een hogere programmeertaal: de compiler plaatst de gebruiker niet voor zulke vreemde verrassingen als bv. een virtual store...)

Ik geloof niet, dat alle aspecten van "complexiteit" met het bovenstaande aan de orde gekomen zijn, ik denk aan de expliciete exploitatie van ons abstractievermogen, zoals ik dat in de kunst van het programmeren probeer te preken. Ik dacht, dat dat een essentieel aspect van de informatica was en dat ik met dat college zo niet een spijker, dan toch wel een spijkertje op de kop heb geslagen. Zowel studenten als leraren uit het hoger beroepsonderwijs hebben zich zeer gunstig over dit college uitgelaten, maar ze reageren verschillend. De leraren uit het hoger beroepsonderwijs zijn er op grond van dit college over het algemeen van overtuigd, dat "programmeren moeilijker is, dan men denkt", maar bovendien, dat, hoewel ik vrijelijk over de kunst van het programmeren praat, het daarmee nog niet in zwarte kunst hoeft te ontaarden en zij begrijpen, dat dit college, met zijn kleine sommetjes, maar een eerste aanzetje is. Studenten hebben de neiging het vak programmeren met sommetjes van deze omvang te identificeren en neigen, mede door de begrijpelijkheid van het college, het vak te onderschatten. Sinds dit college in dictaatvorm is uitgereikt, neemt het aantal zichbij mij meldende afstudeerders ernstig toe. Als dit het gevolg is van een maatschappelijke onzekerheid en men het gevoel heeft, dat men als computer expert nog makkelijker zijn brood zal verdienen -wat ik betwijfel, daar de automatisering een heel kwetsbaar gebied is gebleken, waarin nu een aantal statusprojecten onbarmhartig gedumpt worden-, dan zou Lunbeck deze toevloed ook moeten merken. Ik ben, op grond van bij studenten gepeilde motivering een beetje bang, dat de keuze maar al te vaak een gevolg is van het "succes" van dit college. Ik ben mij er pijnlijk van bewust, dat het vermogen projecten van een orde of twee, drie groter aan te pakken, een essentieel aspect van de informatica is, plus alle denk- en organisatietechnieken om tegen deze schaalvergroting opgewassen te zijn. Het is juist deze schaalvergroting van de problemen, die zo typisch is voor de moderne informatica. Hoe men dit aspect in een curriculum zou moeten recht doen wedervaren is mij een volslagen raadsel.

Ik wil nu twee punten noemen, die mij ernstige zorgen baren.

Het ene punt betreft de effectiviteit van algorithmen. Vele wegen leiden naar Rome en om een bepaald probleem op te lossen, zal men doorgaans de keuze hebben tussen allerlei verschillende algorithmen. Wie is de beste? Ik zie drie verschillende dingen gebeuren. Volgen wij het spoor van Knuth, dan komen wij tot de conclusie, dat afschattingstechnieken, en in het bijzonder allerlei vormen van asymptotiek tot de kern van de wetenschappelijke bagage van de informaticus gerekend moeten worden: hij dient dit stuk wiskunde te beheersen om te kunnen voorspellen, hoe goed zijn overwogen product is. Ik plaats hierbij grote vraagtekens, en wel ten eerste omdat het soort wiskundige technieken, dankzij welke deze afschattingen soms mogelijk zijn, erg probleemgebonden lijken en moeilijk daarom tot de kern van de informatica gerekend kunnen worden, ten tweede omdat de statistische verdeling van de invoer, die het programma zal krijgen toegediend, veelal niet bekend is. De tweede techniek, die van toepassing is, als je het programma inderdaad intensief gebruiken wilt, is "proberen". Het geeft informatica een flavour van een experimentele wetenschap, niet zozeer omdat de programmeur niet weet wat hij gedaan heeft, maar niet weet, wat het programma zal moeten doen. Met de kanttekening, dat die experimentele flavour mij niet aantrekt, teken ik aan, dat het in onderhavige gevallen waarschijnlijk vaak het enig verdedigbare gedrag is. Dit betkent, dat de programmeur zijn programma inderdaad beschouwen moet als lidmaat van een familie naburige programma's, we zitten meteen met het probleem van het afbeelden van programma's op elkaar etc., en het zinvol uitvoeren van deze experimenten, beheerst en tegen een aanvaardbaare prijs en aanvaardbaar risico is zeker een van de kernopgaven van de informatica. De derde consequentie is, [illegible] dat het eigenlijk onmogelijk is om een verdedigbare algorithme te publiceren als de populatie van gevallen niet bekend is; we zien dan ook gepubliceerde algorithmen, die hun rechtvaardiging vinden in hun statistisch gedrag ten opzichte van een heel willekeurig samengestelde verzameling proefgevallen. Ik beschouw dit als een degeneratieverschijnsel.

Er is een tweede ding, dat mij nog veel meer zorgen baart. Ik ben nu ettelijke malen tegengekomen -en zodra je bij bestandsorganisatie, wat je moet, er rekening mee gaat houden, dat er wel eens informatie verloren zou kunnen gaan, krijg je het daar ongetwijfeld ook- dat je pas een fatsoenlijk programma kunt maken, mits je een behoorlijke theorie over de mogelijk plaatsvindende gebeurtenissen hebt geconstrueerd, kortom het soort programma, waarin je hopeloos in de mist komt als je er naief aan begint. Dit verschijnsel kennen we allemaal. We moeten dus tot de onontkoombare [illegible] conclusie komen, dat het tot de typische en essentiele vaardigheden van de competente informaticus behoort, dat hij een dergelijke noodzaak voor een theorie tijdig signaleert. Kan iemand deze ongrijpbare vaardigheid vatten?

In het voorafgaande heb ik me met zorg van specifieke machinetoepassingen onthouden. In principe moeten we natuurlijk ook zeggen, dat als je daaraan begint, het hek van de dam is. Ik vraag me soms af, of we een uitzondering moeten maken voor, zeg maar, administratieve machinetoepassingen. Het is een kwestie van Mohammed en de berg. Het staat vast, dat de meeste machines administratief op een of andere manier worden ingezet, het staat ook vast, dat de mensen, die dit proberen, doo de bank genomen met grote moeilijkheden zitten. Het staat voor mij althans ook vast, dat zo hun problemen oplosbaar zijn, deze redding van een meer mathematische aanpak zal moeten komen. Het probleem is, dat de meeste wiskundigen dit nog niet weten -en het zou daarom misschien makkelijker op zijn te zetten in een apart Department of Computing Sciences dan onder de vleugelen van een afdeling wiskunde-, en dat ook de meeste administratieve machinegebruikers dit nog niet weten. (Sommigen lijken mijn opvatting over de rol van de wiskunde in dezen te delen.)

Vanwege Mohammed en die verdomde berg zou je, als een groep terzake kundige wiskundigen deze nood zien en vertrouwen hebben in hun vermogen hier te helpen, kunnen argumenteren, dat hier voor deze wiskundigen een duidelijke taak is weggelegd. Omgekeerd moet je vermoeden, dat als deze wiskundigen dit gedaan hebben, zij makkelijker aan de administrateurs de relevante opvoeding kunnen overdragen, dan dat de administrateur zijn problemen aan de wiskundige zal kunnen uitleggen. Mijn voorlopige gevoel is dat er ruimte is voor een onderzoek met een sterk wiskundige inslag, dat echter, eenmaal op gang, primair opleiding van derden (c.q. wiskundigen, die een heel speciaal toepassingsgebied in willen) tot gevolg zal kunnen hebben.

Ik heb vanmiddag vijfeneenhalf kantje getikt, ik word wat moe en heb de indruk, dat dit stuk nu langzamerhand lang genoeg is; ik wil de eerstvolgende ciro-vergadering liever niet nog meer uitstellen en het daarom hierbij voorlopig laten.

18 april 1972

prof.dr.Edsger W.Dijkstra

Voor mij concentreert het vak Informatica zich op het beheerst ontwerpen van systemen, die met welgekozen middelen een welgekozen doel moeten realiseren. Ik heb hier tweemaal het woord "welgekozen" en niet het woord "gesteld" gebruikt om tot uiting te brengen

1) dat het niet louter de taak van de informaticus is bestaande -eventueel zelfs: hem opgedrongen- apparatuur te gebruiken, maar dat evaluatie van eventuele apparatuur, de adequaatheidsanalyse, niet alleen tot zijn competentie, maar zelfs tot zijn plicht behoort;
2) dat de ervaring geleerd heeft, dat het zelden verstandig is om het "gestelde" doel zonder discussie te aanvaarden en dat het tenminste de plicht van de informaticus is, om zijn "opdrachtgever" te confronteren met "naburige" doelstellingen, die qua realiseerbaarheid wel eens heel anders zouden kunnen liggen.

Met "ontwerpen" bedoel ik zowel de conceptie van de rekenprocessen als de productie van de bijbehorende algorithmen. Met "beheerst" heb ik willen aanduiden, dat conceptie en productie zo ordelijk zijn verlopen en zo handbaar zijn gedocumenteerd, dat de auteur kan instaan voor de qualiteit van zijn product. En hierbij kan qualiteit zich op een heleboel manieren manifesteren, zoals

1) correctheid
2) aanpasbaarheid
3) overzichtelijkheid
4) voorspelbaarheid in de quantitatieve aspecten van gedrag
5) robustheid t.a.v. mogelijke component malfunctioning
6) efficiency.

Dit gekozen hebbende, dienen we alle relevante technieken, kennis en theorieen te bedenken. Ik wil proberen, die die specifiek zijn voor de informatica en niet direct in een ander vak zijn onder te brengen, eerst te noemen.

1) de techniek van het programmeren met bewuste exploitatie van alle toepasbare abstractiepatronen, de vaardigheid van interfacebewaking
2) de theorie van correctheidsbewijzen
3) de kennis van representatietechnieken
4) de technieken van automatisering van codetransformaties (compilers, assemblers etc.)
5) de kennis van machineeigenschappen, die genoemde transformaties actueel maken
6) de theorieen, die aan dergelijke transformaties ten grondslag liggen.
7) de kennis van notatietechnieken
8) de theorie die vertelt, waarop de bruikbaarheid van notatietechnieken berust.(of die theorie al bestaat, doet nu niet zo terzake)
9) de vaardigheid te detecteren wanneer er een model gevormd moet worden waarvoor een theorie ontwikkeld kan worden en zulks vervolgens op adequate wijze te realiseren
10) de vaardigheid "deelplannen" op adequaatheid en realiseerbaarheid te taxeren
11) organisatietechniek voorzover relevant voor eigen ontwerp- en ontwikkelingswerk. (Ik noem dit als "specifiek voor de informatica" omdat de algemene organisatieleer bij de software ontwikkeling 't er lelijk bij heeft laten zitten.)
12) wat anderen bij doorlezing van dit stuk nog bedenken

Wat hebben de rondom de informatica liggende disciplines hier te leveren? Laat ik met de wiskunde beginnen. Grondslagen, logica, "moderne methoden", een stuk algebra (bv. lattice theory), stochastisch en asymptotisch benul en voorzover we dat onder de wiskunde willen rekenen, perfect taalgebruik. Besturingstheorie mag er ook wel bij, ik weet niet precies, of dat niet onder "systems theory" valt; verder kan ik, bij gebrek aan kennis, over systems theory niet zoveel zeggen.