software, de harde kern van de micro-electronica*

Frank Van der Auwera

PROBLEEMSTELLING

Anno 1981 lijkt de 'vertechnologisering' van de sociale wetenschappen en hun vaktaal, objectief een feit.
Begrippen zoals silicon-chips, RAM, Microprocessoren, pakkingsdichtheid, Large Scale Integration of Digitale Logica, worden uiterst frequent en met traditionele flair aan het (quasi-)wetenschappelijk papier toevertrouwd. Wat vroeger strikt tot het taakdomein van de technische of burgerlijke ingenieur behoorde, houdt sedert de expansie van de Micro-elektronica menige sociologische pen in beweging (Huppes, 1980 ; Zanders, 1980 ; Van Weenen, 1980).

Betreft het hier hoofdzakelijk een déjà-vu uit de late jaren vijftig, toen de automatiseringskoorts respectievelijk automatiseringsangst gevoelig toenam, alvorens onze westerse economie haar 'Golden Sixties' ingleed en de diverse effect-prognoses – zowel de technologie-optimistische als de technologie-pessimistische – slechts partieel met de realiteit bleken te kloppen (Wentink en Zanders, 1980) ?
Daar waar de automatiseringscritici uit deze periode inderdaad niet altijd vrij te pleiten waren van een wat voorbarige 'doomsday' mentaliteit – de technologie zelf werd nog te vaak vanuit een deterministisch perspectief geïnterpreteerd ; er lag nog een ruime afzetmarkt voor productinnovaties braak ; heel wat productieprocessen werden nog niet geautomatiseerd hooguit gemechaniseerd, hetzij door hun complexiteit, hetzij door de nog te hoge omschakelingskosten ; de arbeidsmarkt kon nog rekenen op een redelijke balans tussen de vraag- en aanbodzijde ; ... – maakt de economische crisisachtergrond van dit moment het moeilijk nog onvoorwaardelijk te geloven in een optimistische compensatietheorie (1).

De onvolmaaktheden van ons economisch bestel worden immers met de dag duidelijker : een ontregelde arbeidsmarkt met klimmende werkloosheidscijfers ; een feitelijk failliet van het Sociale Zekerheidsbolwerk ; industrieel een quasi-nulgroei ; een vertroebeld investeringsklimaat ; een economisch herstelbeleid dat zelf voortdurend aan herstelling toe is ; een duizelingwekkende rijksschuld ; een wankelende nationale frank ; een 'shake-out' van 'zwakkere' bedrijven ; ...
Cruciaal probleem tegen deze sombere achtergrond, waarbij termen als (mini-)matiging en (maxi-)versobering schering en inslag zijn, wordt de vraag tot op welke hoogte de Micro-elektronica deze scheefgetrokken relaties, zowel op kwalitatief als op kwantitatief niveau, zal bestendigen dan wel oplossen.
Met het oog op verdere prognoses is het dan ook van het allergrootste belang dat de arbeidssociologie nu reeds rekening houdt met een aantal invloedrijke factoren en

* Bovenstaand artikel werd geschreven in het kader van het F.K.F.O. – project kredietnummer 2.0022.80 "Kwaliteit van de Arbeid" (promotor : Prof. Dr. A.L. Mok).

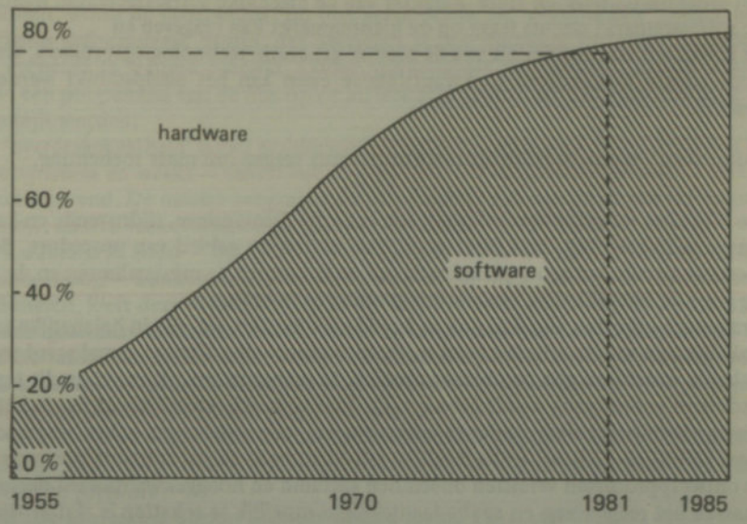
trends, aangezien het verwachte 'revolutionaire' karakter van deze Nieuwe Technologie (2) staat of valt met het reële innovatietempo, in casu de snelheid én accuraatheid waarmee organisaties het medium Micro-elektronica in hun productontwikkeling en productieprocessen (dienstverlening) laten doordringen.
Sleutel op dit toekomstig innovatiepatroon is, naast de beperkingen en weerstanden die vanuit de innoverende organisaties of de betrokken belangenpartijen optreden, vooral de souplesse van de software of programmatuur (3). Deze (micro-)computergrammatica, die de instructies en codes bevat waarmee de apparatuur optimaal moet functioneren, kampt momenteel met een ernstig productiviteitsprobleem. Een productiviteitsvraagstuk, dat de industriële én maatschappelijke impact van de chip aanzienlijk kan verkleinen.

Dit artikel valt dan ook grosso modo uiteen in een dubbele probleemstelling. In een eerste deel wordt de specifieke rol van de "software" in het introductieproces van de micro-elektronica en de praktische knelpunten terzake besproken, terwijl in het daaropvolgende stuk hoofdzakelijk kanttekeningen worden gemaakt bij de strategieën die gevolgd worden om de productiviteitsproblemen in de softwareproductie "op te lossen".

HARDWARE CONTRA SOFTWARE ?

Met de recente commercialisering van de chip (= kleine siliciumschijfjes, waarop steeds grotere aantallen schakelingen worden aangebracht) en de microprocessor (= geheugenbouwsteen met programmeerbaarheid, op chip gemonteerd) heeft de technologie onze industriën ongetwijfeld nieuwe kansen gegeven maar meteen ook nieuwe probleemvelden gecreëerd.

Het prestatievermogen van de (micro-)elektronische bouwstenen ging steeds crescendo, zowel op het vlak van de snelheid, complexiteit, betrouwbaarheid als universaliteit, de prijs van de hardware per geheugencomponent nam voortdurend af (Rathenau, 1979 ; Department of Trade and Industry, 1978 ; Paker, 1979), terwijl de software een kwalitatieve 'surplaçe' beleefde (Duijvestijn, 1975 ; Business week, 1980, Datanews, 1980).
Waar veel geheugencapaciteit wordt aangeboden op steeds kleinere ruimtes en de schakelsnelheid wordt opgevoerd, wijzigt zich ook de prijs/prestatielogica (Nichols, 1976 ; Department of Industry, 1978 ; Zanders, 1980) en stijgt de behoefte aan een adequate programmatuur, die aangepast is aan de krachtiger-geworden microprocessoren en het multi-applicatiekarakter van de Nieuwe Technologie.
In alle opzichten worden we thans geconfronteerd met een tweepolige ontwikkeling : sterkere en goedkopere hardware tegenover soms starre en steeds duurdere software. Deze ongelijke ontwikkeling weerspiegelt zich duidelijk in de onderlinge kostenverhouding tussen de apparatuur (hardware) en programmatuur (software), voorgesteld door de curve van Boehm (Boehm, 1976).



Evolutie van de hardware/software kostprijs in de tijd

Volgens deze stelling staat de kostenrelatie Hardware/Software, die 25 jaar geleden nog van de orde 80/20 was, op het punt volledig om te buigen. Arbeidssociologisch is zulke kostenomkering een interessant gegeven, aangezien het niet langer de apparatuur 'sec' is, die de doorslaggevende factor wordt bij de aankoop van een (micro-)computer, maar de kosten, de flexibiliteit én de accuratesse van het daarbijhorende softwarepakket.
Tweederde van de toegenomen softwarekosten vloeit overigens voort uit het "onderhoud" en de voortdurende aanpassingen van bestaande toepassingsprogramma's.
Voor de industrie stelt de software dan ook een dubbel probleem : kwalitatief, omdat de programma's moeten aansluiten bij de behoeften van de organisatie als commerciële entiteit en bij de kwalificaties en capaciteiten van de directe gebruikers ('user interfaces') ; kwantitatief, omdat de steeds klimmende softwarekosten een zware "lopende" rekening vormen, vooral tegen een achtergrond van een softwarefabricage, die steeds moeilijker aan de toegenomen industriële vraag kan voldoen.

KNELPUNTEN IN DE SOFTWARE-ONTWIKKELING

Op een ogenblik dat software de grendel dreigt te worden op de verdere commercialiseringscyclus van de micro-elektronica, zijn er een aantal "verklarende" factoren aan te wijzen, die de relatieve achterstand van de programmatuur t.o.v. de hardware hebben veroorzaakt, in casu :

1. het arbeidsintensieve en 'dure' karakter van de eigenlijke softwarecreatie, waardoor de softwaremarkt slechts traag op de klantenmarkt kan reageren en
2. de diversiteit en specificiteit van de nieuwe generatie technologiegebruikers, waardoor andere kwalitatieve en kwantitatieve eisen aan het eindprodukt werden gesteld.

De "oorzaken" van deze (tijdelijke) softwarecrisis vragen om meer toelichting.

Primo blijft het schrijven van software een erg arbeidsintensieve, tijdrovende en kapitaalverterende activiteit, met hoge correctheidseisen op gebied van procedure. Softwarecreatie vereist een grote planificatie om de foutenlast te minimaliseren en de tijd nodig voor de ontwikkeling van een programma in te krimpen.
Programmeringskosten zijn dan ook in hoofdzaak loonkosten, die in belangrijke mate afhankelijk zijn van de efficiëntie en snelheid waarmee de software gerealiseerd wordt. Deze beide variabelen zijn vooralsnog moeilijk te manipuleren, alhoewel op dit terrein voortdurend vooruitgang wordt geboekt. De fundamentele moeilijkheid ligt in het feit dat software-ontwikkeling, ondanks jaren ervaring, nog steeds geen echte 'wetenschap' is, maar steunt op de haast ambachtelijke arbeid van een aantal softwareschrijvers (4). Deze ontwerpprocessen vereisten bovendien getraind en hooggekwalificeerd personeel – waarvan het reële vraag- en aanbodaantal soms moeilijk te schatten is, dat in zijn taakuitoefening nog een zekere autonomie en discretie tracht te vrijwaren, gekoppeld aan een vrij hoge salariëring (Business week, 1980).
Menselijke factoren bepalen dan ook in grote mate de limieten van de softwarecreatie. Niet alleen is de feilbaarheid van het menselijk denkproces een beperking bij het programmeren, maar ook de neiging van ervaren programmeurs om nieuwe technieken te verwerpen (Bates, 1976).

Secundo heeft de micro-elektronica, door haar gunstige prijs/prestatieverhouding een uiterst geschakeerde en zelfs gedeeltelijk nieuwe klantenmarkt aangeboord.
De microtechnologie trekt immers ook "kleine" en "ongeoefende" bedrijven als gebruikers aan, die elk specifieke toepassingspaketten en taakeisen formuleren, meestal zonder jarenlange computerervaring.
Vroeger lagen ontwerp, analyse en computergebruik veel meer in elkaars verlengde, terwijl momenteel heel wat potentiële en feitelijke gebruikers computeranalfabeten blijken.
In de praktijk mondt deze nieuwe "computermentaliteit", vooral bij kleine en middelgrote ondernemingen, uit in een erg toenemende vraag naar kant en klare standaardpaketten.
Deze confectieprogramma's (loonberekening, facturatie, ...) vallen in een betaalbare prijsklasse en werken ongetwijfeld tijdbesparend, maar zoals alle standaardprodukten vertonen zij soms een te 'statisch' karakter. Zakenprocedures zijn nu eenmaal voortdurend aan schommelingen of aanpassingen onderhevig, terwijl de werkomgeving van elke organisatie in zekere zin "typisch" is.
Tevens moeten deze programmatiepaketten een hoge graad van gebruiks- en gebruikersvriendelijkheid bezitten, aangezien een deel van het nieuwe klantenbestand niet altijd over de nodige onderlegdheid en kwalificaties beschikt om mathematisch-complexe operaties te verrichten.

Naarmate echter een rijker gamma apparatuur in de diverse industriële productiecycli doordringt, wordt de programmatuur ook een kwestie van 'portabiliteit' ; een in bar nederlands geformuleerd begrip, dat overigens in zijn courante vertaling 'draagbaarheid' eveneens te wensen overlaat. Onder 'portabiliteit' verstaan we dan de mate waarin een programma van de ene op de andere machine en verwerkingsomgeving kan overgetapt worden.
Deze 'overdraagbaarheid' is met andere woorden een maatstaf voor de universaliteit van een programma en werkt – vanuit het oogpunt van het toepassende bedrijf – uiteraard kostenbesparend. De meeste programma's lijden echter aan een grote onverdraagzaamheid t.o.v. andere verwerkingsomgevingen (Duijvestijn, 1975 ; Fenton, 1978).
Deze kwalitatieve eisen – betaalbare, snelgeleverde software met een vlotte mens/machinedialoog – klinken uiteraard niet nieuw in de oren ; zolang er computers functioneren, leeft deze logische vraag in de bedrijfswereld. De micro-elektronica heeft aan deze 'gebruiksvriendelijke' eis echter de dimensie snelheid en tijd toegevoegd : de hardware veroudert sneller i.v.m. vroegere computer-époques ; nieuwe toepassingen en functies stellen voortdurend strengere normen aan de programmatuur ; het profiel van het computercliënteel evolueert vlug ; etcetera ...
Deze veranderde en nog veranderende klantenmentaliteit spreekt ook uit de nieuwe afhankelijkheidsrelatie tussen (eind-)gebruikers en leveranciers/producenten van hardware en software.
Individuele gebruikers blijven dikwijls slechts loyaal t.o.v. een bepaald type (merk, familie) apparatuur omdat het overstappen naar een concurrerend type, belangrijke programma-aanpassingen vraagt of de vertrouwde deskundigheidsbegeleiding van de leverancier doorbreekt, dus onnodige kosten veroorzaakt.
Opnieuw moeten we wijzen op het feit dat, tijdens de vroegere computeriseringsfasen, er een min of meer logisch verband bestond tussen systeemontwikkeling, programmatuuruitbouw en apparatuurgebruik.
Programma's werden veelal 'binnenshuis' gecreëerd of gecorrigeerd bij de toen hoofdzakelijk grote gebruikers.
Hardware en Software werden bovendien als één globaal pakket door de grote hardwarefabricanten aan het cliënteel geleverd.
Toen in 1969 I.B.M., de first lady op de wereldcomputermarkt, opteerde voor een gescheiden prijsbepaling van apparatuur- en programmatuurwerk (Business week, 1980 ; Datanews, 1980), ontstond ook een nieuw type klantenrelatie.
Grootschalig gecomputeriseerde organisaties kampen dan ook dikwijls met een echt "software-dilemma" : ofwel kiezen zij voor een wat eigenzinnige koers (meerdere leveranciers ; problemen van afstemming), ofwel opteren zij voor een 'veilige' klantenrelatie (één leverancier ; 'trouwe' begeleiding). Grof geredeneerd duikt bij grotere microcomputergebruikers wel eens het dilemma op van : administratieve orde màar grote afhankelijkheid of relatieve autonomie maar administratieve verwikkelingen.
De voortdurende en trage aanpassingen van de toepassingsprogramma's maken software echter ook voor de kleinschalige gebruikers tot een quasi-permanente kost, die een bedrijf nogal makkelijk tot loyaliteit dwingt t.o.v. een bepaalde leverancier (Computable, 1981).
Bovendien blijkt uit recente klachten binnen de bedrijfswereld, dat zelfs deze 'begeleiding' vaak te wensen overlaat, daar er tussen de probleemsignalering en probleemoplossing soms een te lange wachttijd verloopt (Computable, 1980).

Grosso modo mogen we dan ook zeggen dat de actuele softwaremarkt duidelijk de sporen draagt van de "intensive technology", die de micro-elektronica toch onmiskenbaar is.
Met de term intensieve technologie, bedoelt men immers de variëteit aan technieken, die kan ontstaan wanneer de keuze, de combinatie en de volgorde van de toe te passen technieken afhankelijk is van de informatie over objecten met sterk veranderlijke eigenschappen (Thompson, 1967).
De onzekerheid waarmee softwareproducenten en kleine of grote eindgebruikers anno 1981 kampen, wijst dan ook op de actuele onmacht bij deze organisaties om 'rationeel' in te spelen op de snel veranderende eisen en mogelijkheden vanuit hun taakomgeving. De manier waarop zij deze 'rationaliteit' strategisch expliciet nastreven vormt het tweede onderdeel van de probleemstelling.

SOFTWARESTRATEGIEEN

Rekening houdend met de in het vorige deel belichte probleemaspecten, wordt de kwestie relevant hoe fabrikanten en hun cliënteel, deze productiviteitscrisis in de software-ontwikkeling kunnen, willen en durven stuiten.
Globaal beschouwd meen ik dat deze softwarestrategieën – het woord 'strategie' staat hier misschien wat voorbarig, aangezien de rationaliteit en intentionaliteit die onmiskenbaar aan het concept verbonden zijn, meestal doorvlochten worden met ad-hoc-beslissingen of postfactum-maatregelen – zich ontwikkelen langs drie hoofdlijnen :

1. het vertalen van de 'softwarecrisis' in termen van opleiding en onderwijs (positie centrale overheid) ;
2. het dichten van tijdelijke aanbodstekorten via een 'externalisering' van de software (groei van de onafhankelijke 'software business') ;
3. het beïnvloeden van de softwareproductie zélf, hetzij via het stroomlijnen van de programmeermethodes, hetzij door een rigieder individuele controle op de softwareschrijver.

Deze drie strategieën, die meestal gecombineerd voorkomen, verdienen echter een nadere analyse.

Bij de *eerste strategie* wordt de softwareproblematiek door producenten en/of bedrijfsleven getild naar het niveau van een maatschappelijk en politiek vraagstuk. Men concludeert dat de link (theoretisch) onderwijs/ (actieve) onderneming, zowel kwalitatief als kwantitatief, onvoldoende wordt gelegd. Het komt de samenleving, in casu de overheid, toe ten allen tijde deze opleidingskloof te overbruggen en te voldoen aan de vraag naar hooggekwalificeerd softwarepersoneel. De slogan "Het onderwijsapparaat is te log en te conservatief om deze kwalitatieve sprong te maken" is dan ook een oud zeer in werkgeverskringen.
Deze benadering impliceert twee belangrijke beleidsaspecten : enerzijds moet de vraag beantwoord worden óf er inderdaad een grote industriële behoefte bestaat aan hooggeschoold softwarepersoneel én welke omvang deze behoefte concreet aanneemt ; anderzijds behelst zij de vraag naar de herom- en bijscholingskansen binnen een bepaald economisch en politiek systeem én de inhoudelijke validiteit resp. bruikbaarheid van de officiële software-opleiding.
Het eerste luik van deze vraag houdt duidelijk een méér kwantitatieve arbeidsmarktdimensie in, terwijl het tweede element eerder de kwalitatieve kennisdimensie omvat. In beide gevallen heeft de literatuur terzake echter het uitzicht van patchwork.
Op het vlak van de arbeidsmarktsituatie, wijst Duijvestijn bijvoorbeeld op het feit dat in de U.S.A. de vraag naar informatiepersoneel jaren terug geweldig steeg, waardoor er een massale omscholing op gang kwam, terwijl nu de vraag opnieuw stagneert en zelfs lichtjes achteruitloopt (Duijvestijn, 1975).

De verklaring voor dit fenomeen in tweeledig :

1. softwarekosten zijn dermate hoog gestegen, dat ondernemingen gezien de loonlast opnieuw op hun informatiepersoneel gaan besparen ;
2. het recessieklimaat 'kelderde' diverse overheidsprojecten (ruimtevaart ; vliegtuigconstructie ; ...), die in de U.S.A. steeds dé steunberen bij uitstek vormen onder elke technologie-ontwikkeling.

Ook Nielen van zijn kant vermoedt dat de werkgelegenheid voor computerpersoneel in het algemeen het profiel van een S-kromme bezit en naar een verzadigingspunt toeloopt. Volgens deze auteur begint de afbraak van het aantal jobs echter aan de voet, wat dus impliceert dat gekwalificeerd softwarepersoneel niet meteen in de directe gevarenzone glijdt (Nielen, 1978).
Het gezaghebbende analysebureau Arthur D. Little echter houdt nog steeds rekening met een blijvende schaarste van programmeurs en hooggekwalificeerd softwarepersoneel. Ondanks het feit dat in de U.S.A. circa 300.000 programmeurs werkzaam zijn én eenzelfde aantal part-time programmeerwerk verricht, blijft – mee onder invloed van de micro-elektronica – de vraag naar hooggekwalificeerde softwareschrijvers het aanbod overstijgen met ongeveer 40 % (Business week, 1980).
De, al dan niet vermeende, discrepansie tussen de vraag- en aanbodcijfers, kan grotendeels herleid worden tot een methodologisch vraagstuk. De meting van deze problematiek berust namelijk grofweg of drie methodes. De eerste methode gaat uit van het aantal geïnstalleerde computers, waarna het gemiddelde berekend wordt van het aantal functionarissen die met deze computers werken. Hierbij neemt men bovendien een specifiek groeiritme van het computerbestand aan.
De tweede methode maakt bij de berekening simpelweg gebruik van enquêtering, terwijl de derde manier ("de spiegelmethode") veronderstelt dat West-Europa een identieke evolutie zal doormaken als de U.S.A., mits enige jaren vertraging (Duijvestijn, 1975).
Elk van deze drie methoden zorgt voor een zekere vertekening, zoals Duijvestijn in zijn artikel illustreert, wanneer men de gemiddelde personeelsbehoefte per installatie bekijkt in een aantal commissierapporten (Duijvestijn, 1975).
Eenzelfde divergentie manifesteert zich op het vlak van de 'kwalitatieve' aspecten, die betrekking hebben op de (in)adequaatheid van het gangbare opleidingsmodel voor softwarepersoneel.
Hier wordt voornamelijk met het argument geschermd dat de software-opleiding – en de opleiding van informaticapersoneel in breedste zin – té theoretisch geïnterpreteerd wordt en zelden praktijktraining tijdens de onderwijsperiode inbouwt.

Kandel wijst er bijvoorbeeld op dat Amerikaanse universiteiten te zwaar het accent leggen op de louter theoretische aspecten van de informatica, zoals automatentheorie, Booleaanse algebra, tralies, semigroepen en universele algebra's, waardoor gegradueerde studenten géén echte problem-solving aankunnen (Kandel, 1972).
Indentieke geluiden hoort men eveneens in West-Europa (Rathenau, 1980 ; Computable, 1980). Opvallend is echter de uitspraak van de Onderzoekscommissie Rathenau, die – alhoewel zij goochelt met begrippen als permanente vorming en omscholing onder invloed van de chiptechnologie – tóch onvermijdelijk de vinger in de wonde draait wanneer zij concludeert : "Kennis verschraalt en het lijkt waarschijnlijk dat de introductie van micro-elektronica dit verschralingsproces zal versnellen." (Rathenau, 1980).
Op dit punt situeert zich inderdaad de hamvraag van de afstemmingsproblematiek Onderwijs/Bedrijfswereld : wat baat het de arbeidsmarkt via het onderwijssysteem 'schatplichtig' te maken aan een bedrijfswereld met haar (tijdsgebonden, dynamisch) behoeftenpatroon, wanneer de 'vereiste' kennis door toedoen van snelle technologische en wetenschappelijke evolutie, elke vijf à tien jaar 'veroudert' (de-kwalificatie) ?
De vicieuse cirkel van omscholing, bijscholing, herscholing, zelfstudie en permanente vorming, kan misschien wel een theoretisch-utopisch aantrekkelijke visie zijn, maar beleidsmatig bevat zij nog heel wat vraagtekens zoals : wie moet deze massale training financieren ? wie dirigeert de inhoudelijke aspecten van de opleiding ? hoe kan de dynamiek van bedrijfsspecifieke behoeftes ooit in een opleidingssysteem geprojecteerd worden ?

De *tweede strategie* bestaat erin dat bedrijven, die mini- of microcomputers in hun machinepark introduceren, hun software verder 'externaliseren'. Het uitbesteden van software-arbeid en het versneld aanschaffen van standaardpaketten is een logisch gevolg van een bedrijfsbeleid dat de loonkost, verbonden aan programmatuurontwikkeling en systeemanalyse, zoveel mogelijk wil drukken.
Deze trend naar 'externe' softwarebronnen, naast de bestaande contacten met de fabrikanten van bepaalde apparatuur, kreeg pas goed de wind in de zeilen toen I.B.M. in 1969 startte met een ontdubbelingspolitiek, waardoor softwarebegeleiding een winstgevende zaak werd en er weldra een "independent software business' ontstond : een sector die functiegerichte begeleiding geeft op het vlak van analyse en programmeren (softwarebureau's, systeemhuizen, ...), via het streven naar octrooirecht op programmatuur, het geven van bijscholing, het bestuderen en stimuleren van de exportmarkt, het verkrijgen én onderhouden van contacten met de overheid, ... (Datanews, 1980 ; Business week, 1980 ; Computable, 1980) (5).
Deze nog jonge industrie, amper tien jaar actief, komt nu pas in volle bloei, dankzij de 'bredere' aanvaarding van de micro-elektronica. Vooral in Nederland – waar deze ontwikkeling reeds geruime tijd aan de gang is, terwijl men in België pas onlangs de Nieuw Technologie als 'probleemkind' ontdekt heeft – neemt men heel wat beweging in deze 'onafhankelijke' softwaresector waar, vooral qua activiteitenbundeling. (Computable, 1980). Het blijft echter de vraag welke marktaandeel dit 'onafhankelijk' softwarecircuit zich in de nabije toekomst weet toe te eigenen, gezien het feit dat de grote hardwareproducenten de programmaproductwinsten hebben geroken en hun verkoopsstrategie hierop afstemmen.
Getuige van deze nieuwe mentaliteit is nogmaals koploper I.B.M., waar blijkbaar een

specifieker programmaverhurings- en dienstverleningsbeleid op punt werd gesteld (Datanews, 1981a).

Vanuit arbeidssociologisch perspectief, biedt de *derde strategie* misschien wel de boeiendste stof. Hardwarefabricanten, die zich met volle gewicht op de softwaremarkt hebben geworpen en 'grotere' computergebruikers, die zich in de luxe van een – volledige of gedeeltelijke – softwareproductie binnen de eigen fabriekspoorten of kantoordeuren kunnen veroorloven, willen de productiviteitscrisis in de software-ontwikkeling op twee manieren beheersen.
De eerste manier ligt voor de hand. Men kan trachten het programma-ontwerp – als techniek én procesmatig gebeuren – verder te formaliseren, voorspelbaar te maken en te 'disciplineren'.
Rond 1970 werd het "Structured Programming" of "Structured Analysis" dan ook als een deus-ex-machina in de bedrijfswereld onthaald, alhoewel deze methode een decade later nog steeds niet de voorspelde 'sensationele' omwenteling heeft teweeg gebracht. Niettegenstaande het oorspronkelijke begrip "gestructureerd programmeren" de laatste jaren met talrijke organisatorische en methodologische aspecten werd verzwaard, waardoor er een uiterst onduidelijke situatie is ontstaan (Bates, 1976), kunnen we het toch als volgt omschrijven :
"The chief instrument in industrializing software production has been structured programming. Briefly, structured programming stresses orderliness, simplicity, and economical use of standard language and code. It limits the range of choices available to programmers in designing and writing a program. Instruction sequences are specified, limited, or prohibited altogether, Some structured techniques even limit the number of instructions a programmer may use in a program or "subroutine". The latter are now commonly called "modules" and these in turn allow futher fragmentation of software tasks. Modules – which are simply discrete components of a larger program – can be parcelled out to a low-level programmer who must follow rigid coding guidelines in writing the program fragment. He or she no longer needs to know anything about the overall system of which it is a part, or even how the module fits into other modules." (Kraft, 1979).

Kraft noemt "Structured Programming" dan ook het antwoord van de softwaremanager op de assemblagelijn, aangezien minder moet vertrouwd worden op individuele, hooggekwalificeerde softwarewerkers en er een verregaande taakfragmentering in het programmeerwerk mogelijk is geworden.
Van zijn kant wijst De Kerf erop dat rond deze gestructureerde methode een hele cultus is ontstaan en dat kritiek hierop door sommige kringen wordt verketterd (De Kerf, 1979). Wanneer men echter vluchtig de literatuur onder ogen neemt, kan men besluiten dat niet iedereen het "Structured Programming" als een verloren zaak beschouwt (Wellekens, 1979 ; Slater & Modell, 1978 ; Chapin & Denniston, 1978 ; Jordan & Urban, 1978 ; ...). De moeilijkheden om programmeerwerk te plannen en te (her)structureren, zijn echter in belangrijke mate van menselijke aard, aangezien een aantal "psychologische" condities dit planificatieproces determineren (Weinberg, 1971 ; Hoc, 1979).
Het tweede onderdeel van deze 'beheersingsstrategie' bestaat erin de individuele softwareschrijvers bij hun taakuitoefening nauwlettender te controleren, aangezien zij de

kwetsbaarste schakels vormen in de ontwerpketen. Wil men efficiëntie en productiviteit opdrijven, dan zal men de individuele prestaties moeten meten, eventueel normeren. Gestructureerd programmeren én maatprogramma's waren reeds de eerse stappen in dit "routiniseringsproces", aangezien zij een deel van de kwalificaties van het softwareschrijven aftapten (Kraft, 1979).
Opnieuw geeft computerreus I.B.M. de trend aan. Enerzijds zien we het inschakelen op grote schaal van (hulp)computers bij het programmeer-ontwerp, waardoor de programmacomposities van de 'technicus' vrij snel door de machine van hun foutenlast worden gezuiverd of de coderegels worden gesimplificeerd (Westley, 1979).
Bij I.B.M. staat deze aanpak borg voor een verdubbeling van de productiviteit (van 30 tot 60 lijnen per dag ; minder fouten ; makkelijker te reviseren) en met de ontwikkeling van de "Application Development Facility" lijkt deze trend aan te houden. Deze programmeerhulp stelt een programmeur met minimum-skills in staat, na opgave aan de terminal van de kenmerken van het gewenste systeem, automatisch de vereiste codering uit de computer te krijgen. Het schrijven van een programma wordt op die manier herleid tot één honderdste van het aantal codes, dat bijvoorbeeld in COBOL zou nodig zijn (Business Week, 1980 ; Datanews, 1980).
Er ontstaat met andere woorden stilaan een nieuw type 'arbeidsverdeling' tussen (hulp)apparatuur en softwareschrijver : on-line terminals vervangen pen en papier – jarenlang het 'wapen' in handen van gekwalificeerde programmeurs – en nemen geleidelijk de 'craftmethode' van de softwarevakman over, wat een dreigende de-kwalificatie voor deze laatste veronderstelt. Anderzijds ontdekken we een toenemende druk op de softwareprofessie, namelijk de instelling van een 'librarian' bij I.B.M. Deze verslaggevers die in opdracht van het management aan teams van software-ontwerpers en programmeurs worden toegevoegd, houden een dagboek van de activiteiten in de afdeling bij, waarin naast de definitieve coderegels van het programma ook de (individuele) 'foutieve' regels en tussenstapjes worden geregistreerd (C.S.E., 1980). De mogelijke gevolgen, van zulke interne controle, lijken talrijk : software-specialisten kunnen, zowel individueel als in teamverband, voor hun (in)efficiëntie ter verantwoording geroepen worden ; spanningen in de onderlinge werkrelaties kunnen toenemen, omdat het rendementsidee de groep binnesluipt ; theoretisch bestaat de kans dat het loopbaanperspectief van de programmeurs in kwestie gekoppeld wordt aan deze (on)gunstige verslaggeving.
Deze beide ontwikkelingslijnen – verlies van een stuk taakbeheersing aan de hulpcomputer en de controle op de taakuitoefening door teamwaarnemers – brengen ons reeds dicht bij de idee van "software factories" (Business week, 1980 ; Datanews, 1980) : softwarefabrieken, waar op grond van Scientific Management en (neo)tayloristische principes, de programmeertaak en taalconstructie maximaal wordt geautomatiseerd en aan een intensieve supervisie onderworpen, ter bevordering van de globale productiviteit en snelheid. Met andere woorden de stijgende vraag naar complexe en betrouwbare software, schijnt – "paradoxaal" genoeg – historisch hand in hand te gaan met een toenemende inperking van het softwareschrijven als autonome professie (Kraft, 1979).

SLOTBESCHOUWINGEN

Uit voorgaande pagina's mag blijken dat software in toenemende mate de Achillespees van de micro-elektronica – en computertechnologie globaal – dreigt te worden, maar dat de softwareproducenten alles in het werk stellen om de 'productivity-gap' zo snel mogelijk te overbruggen. De grotere beschikbaarheid van geheugencapaciteit voor een breder netwerk van gebruikers, met hun specifieke eisen, heeft dit vraagstuk de afgelopen jaren extra profiel gegeven.

De evolutie van de software zélf, blijft echter afhankelijk van drie factoren, in casu :

1. de mate waarin producenten erin slagen het programma-ontwerp verder te plannen, structureren en te beheersen, vooral via inschakeling van hulp-apparatuur ;
2. de mate waarin de micro-elektronica hardware zich kwalitatief weet te ontwikkelen en kwantitatief te verspreiden, zowel in de industriële sector als in de dienstverlenende organisaties ;
3. de mate waarin zich een nationaal en internationaal innovatiebeleid kan articuleren.

Vooral dit laatste punt vormt een conditio sine qua non voor de competiviteit van de economie in de E.E.G.-lidstaten op wereldvlak, die zich moeilijk kunnen handhaven op de uiterst dynamische informatica wereldmarkt (Commissie van de Europese Gemeenschappen, 1979 ; Datanews, 1981), tenzij op het terrein van de telecommunicatie (1/3 van de wereldmarkt).
Wanneer software in de komende jaren inderdaad 80 à 90 % van de totale investering gaat bestrijken van een informatiseringsproces, loont het vanuit de nationale overheden zeker de moeite om de binnenlandse softwarecreatie te stimuleren om op die manier de technologische afhankelijkheid t.o.v. vreemde fabrikanten in te dijken. De strijd om de hardwareproduktie lijkt voor landen als België momenteel toch een verloren zaak.

Al deze bedenkingen dikken dan ook het vermoeden aan, dat de zo vaak bejubelde of verguisde "Decade van de Chip" opnieuw correcter moet geformuleerd als de "Decade van de Software", waarbij de economische kaarten tussen naties, regio's, sectoren en organisaties andermaal hardhandig geschud zullen worden.

SAMENVATTING

In volle (sociaal-)wetenschappelijke discussie omtrent de korte en lange termijn impact van Micro-Elektronica op arbeid en tewerkstelling, staat de vraag centraal welke factoren het introductie- en adaptatiepatroon van deze nieuwe technologie in ons bedrijfsleven determineren.

Belangrijke grendel op de commercialiseringscyclus van de 'chip' is de software of programmatuur. De snelheid waarmee de hardware is geëvolueerd, werd niet gecompenseerd door een gelijkaardige kwalitatieve en kwantitatieve groei in de software, waarvan het kostenaandeel in het totale innovatiepakket (hardware, exploitatie, software, ...) echter aanzienlijk toegenomen is.

Dit artikel tracht daarom enerzijds aan te duiden waarom de software momenteel een 'surplace' doormaakt en beschrijft anderzijds hoe ondernemingen deze productiviteitscrisis in de softwareontwikkeling strategisch willen of kunnen 'bijsturen'.

ABSTRACT

In the height of the (socio-)scientific discussion concerning the short and long term impact of Microelectronics on work and employment, one question occupies a central position : which factors determine the introduction and adaptation pattern of this new technology in our industries ?

Software constitutes an important bolt on the chip's commercialization cycle. The speed with which hardware evolved, was not compensated by a similar quantitative and qualitive growth in software, whose share in the totality of innovation costs (hardware, daily use, software, ...) nevertheless increased considerably.

This article attempts to prove on the one hand why software experiences a temporary delay in its evolution and it describes on the other hand which 'strategies' companies wish to bring into play, in order to meet the productivity crisis in the software development.

NOTEN

1. Neo-klassieke theorieën m.b.t. innovatie stellen dat de technologische ontwikkeling, omwille van haar kostenbesparend effect, leidt tot prijsdalingen die de globale vraag stimuleren, waardoor ook de vraag naar arbeid toeneemt en nieuwe jobs gecreëerd worden. Met andere woorden : de eventuele negatieve effecten op korte termijn (bv. werkloosheid in specifieke jobs ; sectoren ; categorieën ; ...) worden door een toenemende economische spankracht, op langere termijn quasi-automatisch gecompenseerd door het "mechanisme" van de technologische vernieuwing zélf.
2. Gezien de enorme diversiteit in de reeds gerealiseerde en nog mogelijke toepassingen van de micro-elektronica, wordt soms het wat pompeuze begrip "Nieuwe Technologie" in deze tekst gebruikt. Dikwijls blijkt het immers moeilijk om "de" micro-elektronica als een geïsoleerde variabele te behandelen, omdat chiptechnologische toepassingen worden aangewend naast en/of gemengd met meer conventionele apparatuur.
   Het begrip "Nieuwe Technologie" lijkt in die gevallen meer plausibel.
3. Onder de term software wordt doorgaans begrepen het ontwerp en de gedetailleerde sequentie van programma-instructies, die de hardware of apparatuur (transistoren ; microprocessoren ; ...) optimaal laten functioneren. Historisch is deze conceptuele scheiding hardware/software echter regelmatig aan begripsexpansie of -sanering toe (KRAFT, 1979), en merken we een steeds toenemende substitutie van traditionele hardware door software.
4. Opnieuw moet gewaarschuwd worden voor een zekere begripsvertekening. In deze tekst wordt om praktische redenen het software- en informaticapersoneel als een koepelbegrip benaderd, terwijl het in concreto gaat om een meer gedifferentieerde populatie (bv. analysten versus programmeurs).
   In het bestek van de behandelde probleemstelling, lijkt deze brede benadering echter niet onoverkomelijk.
5. De informatica-dienstenmarkt is actief op vier niveau's :
   1. het service-bureau (verhuurt programmatuur en computertijd) ;
   2. het informatica-uitzendbureau (verhuurt personeel voor vastgestelde termijn) ;
   3. het klassieke softwarehuis (vertaalt specifieke organisatieproblemen in programmatuurtermen en het toepassen hiervan op computers van de klant) ;
   4. het systeemhuis (aanvaardt de totale verantwoordelijkheid van een automatiseringsproject, nl. probleemanalyse, de toepassing en de opleiding).

   Vooral op niveau 3 en 4 dient men een onderscheid te maken tussen afhankelijke en onafhankelijke software- en systeemhuizen. Het "afhankelijke" huis sluit een overeenkomst met 1 of 2 constructeurs, het "onafhankelijke" daarentegen heeft geen overeenkomst of een lopend contract bij alle constructeurs. (Datanews, 1981c).

BIBLIOGRAFIE

BATES D. (ed.), "Structured Programming", *Infotech*, State of the Art Report 27, Maidenhead, 1976.

BOEHM B.W., "Software Engineering and Life Cycle Planning", *Datamation*, December, 1976.

BUSINESS WEEK, "Missing Computer Software", *Business week*, Special Report, 1980, pp. 46-53.

CHAPIN N. & DENNISTON S.P., "Characteristics of a Structured Program", *Sigplan notices*, nr. 5, 1978, pp. 36-45.

COMMISSIE VAN DE EUROPESE GEMEENSCHAPPEN, "De Europese Maatschappij ten aanzien van de Nieuwe Informatietechnieken : een communauteir antwoord", *COM 79/650 DEF*, 1979.

COMPUTABLE, "Poging tot vorming van 2e Vereniging van Softwarebureau's", *Computable*, 10 september 1980, p. 17.

COMPUTABLE, "Automatisering gaat bij kleine bedrijven nog niet vanzelf", *Computable*, 17 juli 1981, pp. 1-2.

CONFERENCE OF SOCIALIST ECONOMISTS, *Capitalist Technology and the Working Class*, London, 1980.

DATANEWS, "Overvloed aan Hardware, maar nijpend gebrek aan programmatuur", *Datanews*, nr. 23, 9 december 1980, pp. 10-11.

DATANEWS, "Een Nieuw beleid bij I.B.M. ?", *Datanews*, nr. 2, 20 januari, 1981a, p. 6.

DATANEWS, "Chip en de Leeuwentemmer", *Datanews*, nr. 2, 20 januari, 1981b, pp. 12-13.

DATANEWS, "De programmatuurzaak : 4 trappen naar de software-hemel", *Datanews*, nr. 53 maart 1981c, pp. 11-12.

DE KERF J., "APL – Another Programming Language", *Informatie*, nr. 1, 1979, pp. 35-52.

DEPARTMENT OF INDUSTRY, *"Micro-Electronics : The New Technology"*, London HMSO reprint, 1980.

DUIJVESTIJN A.J.W., "Dertig Jaar Informatice : Hoe zal het verder gaan ?", *Informatie*, nr. 6, 1975, pp. 306-315.

FENTON J.S., "Overdraagbaar maken van programma's : 'n praktijk probleem", 4 delen, *Computable*, nrs. 7/8/9/10, 1978.

HOC J.M., "Le problème de la planification dans la construction d'un programma informatique", *Le travail humain*, nr. 2, 1979, pp. 245-260.

HUPPES T., *Maatschappelijke gevolgen van 'chip' technologie*, Stenferd Kroese, 1980.

JORDAN W. & URBAN H., *Strukturierte Programmierung*, Berlin, 1978.

KANDEL, artikel in COMMUNICATIONS A.C.M., juni 1972, geciteerd in *Duijvestijn*, o.c., 1975, p. 313.

KRAFT, P., "The Routinizing of Computer Programming", *Sociology of Work and Occupations*, vol. 6, nr. 2, may 1979, pp. 139-155.

LEYDER, R., "De computer, de werkgelegenheid en de crisis", *Informatie*, jrg. 21, nr. 7/8, juli/augustus 1979, pp. 408-426.

NICHOLS A.J., "An Overview of Microprocessor Applications", *Proceedings of the IEEE*, juni, 1976.

NIELEN G.C., "Filosofische kijk op toekomst van onze automatisering", *Computable*, 20 januari, 1978.

PAKER Y., "Technologische Ontwikkelingen op het gebied van mini en microcomputers", *Informatie*, nr. 9, 1979, pp. 510-516.

RATHENAU G., *Rapport van de Adviesgroep "Maatschappelijke Gevolgen van de Micro-Electronica"*, Den Haag, Staatsuitgeverij, 1980.

SLATER W. & MODELL H., "Structured Programming Considered Harmful", *Sigplan notices*, nr. 4, april 1972, pp. 76-79.

THOMPSON, J., *Organization in Action*, McGraw-Hill, New York, 1967.

VAN DER POEL W.L., "Software voor Microprocessoren", *Informatie*, nr. 6, 1979, pp. 367-372.

VAN WEENEN B., "Gevolgen van automatisering in de administratieve sector", *Mens en onderneming*, 1980, p. 12.

WEINBERG G.M., *The Psychology of Computer Programming*, New York, 1971.

WELLEKENS W., "Een macroschema techniek voor gestruktureerd programmeren", *Informatie*, 1979, pp. 729-738.

WENTINK A. & ZANDERS H., *De Chips in de Jaren Tachtig*, VIFKA, 1980.

WESTLEY A.E. (ed.), "Software Testing", *Infotech* State of the Art Report 49, 2 vol., 1979.

ZANDERS H.L.G., "Chips, Automatisering en Arbeid", *Sociale Wetenschappen*, nr. 2, 1980, pp. 81.118.

rubriek lopend onderzoek

Met dit nummer biedt de redactie een vijftigtal aankondigingen van lopend onderzoek in Vlaanderen, met een doorbraak van de niet-universitaire onderzoekscentra. De redactie hoopt hiermee een gelegenheid te scheppen om de communicatie tussen onderzoekers te bevorderen.

Instellingen, diensten, onderzoeksbureaus, die nog niet vermeld zijn kunnen zich aanmelden op het redactie-adres teneinde het informatie-net uit te breiden, zodat deze service van het TVS nog relevanter wordt.

Kortelings zal een nieuwe informatieronde georganiseerd worden.

Voor de redactie

Erik Henderickx (RUCA)
& Ivo Nuyens (KUL)