compact

COMPUTER EN ACCOUNTANT

Klynveld Kraayenhof & co
ACCOUNTANTS

NUMMER 17 6E JAARGANG VOORJAAR 1979

compact

VAN DE REDACTIE

Dit voorjaarsnummer van 1979 wordt geopend met een artikel over de invloed van de invoering van minicomputers op de accountantscontrole, door J.F.C. van Epen.
De artikelserie over de Zeefmethode door C. Rietveld wordt in dit nummer besloten.
In de serie "Het gebruik van de computer in de accountantscontrole" dit maal een bijdrage van A. van der Drift over Combi, een testprogramma dat reeds jaren geleden door Klynveld Kraayenhof & co is ontwikkeld.
De vaste rubrieken ABC-Nieuws - deze keer onder andere met een behandeling van de IBM 8100 minicomputer en een $ 10.200.000 fraude - en Literatuuroverzicht - waarin een vijftal boeken worden aanbevolen - sluiten dit nummer.

Compact is een uitgave van de groep Automatisering en Controle van Klynveld Kraayenhof & co.

Het doel van deze uitgave is informatie te verstrekken over ontwikkelingen op het gebied van automatisering en controle in binnen en buitenland.

Deze informatie is in de eerste plaats bestemd voor diegenen, die in de algemene controlepraktijk werkzaam zijn.

Redactie:
A.W. Neisingh, J. Philippo en D. Steeman.
Adres: Pr. Irenestraat 59, Amsterdam

DE BETROUWBAARHEID VAN DE ADMINISTRATIEVE GEGEVENSVERWERKING MET BEHULP VAN MINICOMPUTERS

door J.F.C. van Epen

I Inleiding

Wanneer we ons willen gaan richten op de vraag welke maatregelen van beveiliging moeten worden getroffen bij toepassingen met behulp van minicomputers, dienen we eerst te bepalen wat nu de reden is dat deze toepassingen speciale aandacht moeten krijgen.
Vier redenen wil ik noemen zonder daarbij te pretenderen volledig te zijn:

1. De organisatie is anders: geen rekencentrum.
2. De hardware en standaardsoftware zijn beperkter uitgevoerd.
3. De mogelijkheden welke in de applicatieprogrammatuur kunnen worden opgenomen zijn sterk afhankelijk van de mogelijkheden van de beschikbare programmeertalen. En bovendien...
4. Het gebruik van kleinere computers neemt snel toe.

Het Financieele Dagblad publiceerde 7 oktober 1978 een tabel van een schatting van het aantal geplaatste installaties per 1 januari 1978. Hierin zijn niet de zogenaamde officecomputers opgenomen.
Op een totaal van 10.264 in Nederland geplaatste installaties blijken "slechts" 1.840 stuks (= 17,9%) mainframes te zijn. 17,2% zijn small business computers (aantal ten opzichte van een jaar eerder meer dan verdubbeld!) en de resterende 64,9% bestaat uit mini's. Welk deel hiervan voor administratieve toepassingen in gebruik is, is niet bekend. Wel wordt vermeld dat ca. 2.200 stuks als intelligente terminal dienst doet.
We kunnen er gerust van uit gaan, dat er meer minisystemen dan mainframes voor administratieve werkzaamheden in gebruik zijn, zij het uiteraard voor minder omvangrijke toepassingen.

Ten gevolge van de voortschrijdende technologische ontwikkelingen wordt de afbakening van de verschillende typen hardware, zoals het onderscheid tussen mainframes, mini- en officecomputers moeilijker Er komen steeds meer kenmerken van het ene type ook bij andere typen voor.
Een mogelijke typering is een minicomputer te omschrijven als een "centrale eenheid met besturingssysteem waaraan een breed scala van periferie kan worden aangesloten". Een officecomputer kan worden getypeerd als een "te zamen met de randapparatuur gesloten eenheid zonder besturingssysteem". Deze laatste categorie wordt ook aangeduid met "Visible Record Computer" (VRC).

Vanuit de minitechniek is vervolgens de "Small Business Computer" ontwikkeld. Dit is een vaste configuratie voor kantoortoepassing, waarbij de kern meestal uit een minicomputer bestaat. Het besturingssysteem is hier doorgaans aangepast aan de specifieke eisen die een administratie stelt. Een dergelijke computer biedt ten opzichte van de officecomputer een aantal voordelen, zoals meer intern

geheugen, snellere verwerking, gemakkelijker programmeerbaar. Kortom: hij heeft meer gelijkenis met zijn "grote broer". En ten gevolge van de aanhoudende prijsdalingen in de elektronische industrie is het grote voordeel van de officecomputer, namelijk de lage prijs, achterhaald.
Met de opkomst van de microprocessor is deze prijsdaling zodanig snel gegaan, dat voor uiterst geringe prijzen reeds aardige computertjes worden gefabriceerd, de "Hobbycomputer". Deze ontwikkeling zou hier niet zijn vermeld, ware het niet dat op de vorig jaar gehouden efficiencybeurs zo'n hobbycomputer werd gedemonstreerd waarop reeds een administratief pakket (voor de kleine, middenstandsonderneming) was geïmplementeerd.

Welke voordelen worden nu aan deze kleinere systemen (van hobbycomputer tot small business computer met mainframe-achtige eigenschappen) toegekend, dat zij in korte tijd een aanzienlijk marktaandeel wisten te veroveren?

1. De gebruiker staat dichter bij de computer, waardoor een grotere inzet van hem verwacht mag worden.
2. Ten gevolge van de beperkingen van de apparatuur, die echter steeds minder worden, worden de toepassingen in kleinere eenheden ontwikkeld: men is gedwongen tot "modulaire" systeemopzet. Vele pakketten zijn ook volgens dit principe geconstrueerd.
3. De organisatie rond het gehele automatiseringsgebeuren kan kleiner worden gehouden.
4. Goedkoper!

Een groot nadeel komt vooral voort uit het laatste punt, dat de wens impliceert om "het geheel", goedkoop te houden, ten gevolge waarvan de risico's worden onderschat. Vooral in een mini-omgeving dienen hardware, besturingssoftware, programmatuur en organisatie op elkaar afgestemd te worden.
Dit wordt vaak vergeten. Wanneer namelijk niet aan de betrouwbaarheidseisen met betrekking tot één dezer componenten kan worden voldaan, dienen elders compensaties gevonden te worden. Bijvoorbeeld in eigen programmatuur of in de organisatie.
Onze eis aan het totale systeem (apparatuur, programmatuur en organisatie) luidt niet anders als voor mainframe-toepassingen: Volledige, juiste en ongestoorde verwerking dient gegarandeerd te zijn.
We hebben echter te maken met beperkingen in de organisatie, de apparatuur en de standaardsoftware.

Voor de beoordeling van de aspecten van beveiliging speelt de hardware een enigszins ondergeschikte rol. Van belang is WAAR en HOE en WAARVOOR de computer wordt gebruikt.

WAAR?	Uitgangspunt voor onze benadering is dat de computer op lokatie, dit wil zeggen in de organisatie van de gebruiker is geplaatst. Een rekencentrum met een daarbij behorende zelfstandige organisatie ontbreekt.
HOE?	De gebruiker bedient, hetzij in de vorm van data entry/ batch update, hetzij in vraag en antwoordvorm (interactief) de apparatuur, soms ook in een online/real-time situatie.
WAARVOOR?	De computer wordt gebruikt voor administratieve doeleinden; er wordt tevens gebruik gemaakt van gegevensbestanden.

Hieruit blijkt dat ook de terminal die is aangesloten op een time-sharing service, evenals de terminal aangesloten op een intern mainframe tot het onderwerp van deze studie behoort, mits de verantwoordelijkheid voor de produktie geheel is gelegen bij de gebruiker.

Deze inleiding kan worden afgesloten door te stellen dat de automatisering met behulp van kleinere computersystemen de aandacht van de accountant verdient, zodra de computer gegevens oplevert die vallen onder de verklaring van de accountant. Hoewel de controle "om de computer heen" kan plaatsvinden, dient hij er zich wel van te overtuigen dat de te controleren gegevens te allen tijde door de computer opgeleverd moeten kunnen worden, dan wel reconstrueerbaar zijn, alsmede dat de continuïteit van de gegevensverwerking niet in gevaar komt.
Al deze computersystemen, hoe groot of hoe klein ook, zullen we in de verdere loop van dit verhaal betitelen als "Kantoorcomputers".

Uit het vorenstaande blijkt dat het onderwerp van dit artikel als volgt nader kan worden gepreciseerd:
De betrouwbaarheid van de administratieve gegevensverwerking in organisaties die daarvoor op lokatie in eigen beheer gebruik maken van computerfaciliteiten.

De volgende aspecten zullen hierbij, al dan niet expliciet, aan de orde komen:

- organisatie,
- hardware,
- verwerking
- bestanden.

II Organisatie

In de inleiding is reeds naar voren gekomen dat de toepassingen welke wij beschouwen in organisaties van kleine omvang zijn te vinden. Gesteld is dat het rekencentrum als zelfstandige organisatie ontbreekt. Dit impliceert dat een aantal taken die in een dergelijke organisatie plegen te worden uitgevoerd, nu door de gebruikersorganisatie zullen worden vervuld. We noemen taken als produktieplanning, bestandsbeheer, bestandsbeveiliging, beheer programmabibliotheek,

produktiecontrole.
De gebruiker zal deze taken dienen over te nemen. Zij zijn echter voor deze gebruiker ondergeschikt aan zijn hoofdtaak, het voeren van een administratie.
Het gevaar is aanwezig dat verwaarlozing van een of meerdere taken plaatsvindt, vooral op momenten dat de hoofdtaak alle aandacht opeist. Een ander gevaar is dat de noodzaak van het vervullen van dergelijke taken in het geheel niet wordt gevoeld.
Een en ander kan worden verbeterd door binnen de gebruikersorganisatie een functionaris verantwoordelijk te stellen voor het automatiseringsgebeuren (de computer- of systeembeheerder). Hierdoor wordt binnen de gebruikersorganisatie de noodzakelijke functiescheiding tussen beheer en bewaring/verwerking van gegevens gecreëerd. Voorwaarde is uiteraard dat de systeembeheerder niet tevens een gegevensbeheerstaak vervult.
In hoofdstuk IV zullen nog enkele suggesties aan de hand worden gedaan tot beheersing van de automatisering in de kleine organisatie.

III Invoer van gegevens

We zullen aandacht moeten schenken aan de specifieke eigenschappen van de computer, waarop de verwerking zal gaan plaatsvinden. Hiertoe behoren ook de eventueel "niet aanwezige zekerheden" (men denkt zo snel dat het "wel goed zit"). We noemen:

- Bestaat er zekerheid dat een ingetoetst of ingelezen gegeven niet ongemerkt kan verminken?
- Zijn gebruikersgegevens en machinebesturingsgegevens van elkaar zodanig te onderscheiden (bijvoorbeeld door een geheel afwijkend bitpatroon), dat bij foutief lezen geen verwisseling kan ontstaan?
- Zijn we er zeker van dat de in het geheugen aanwezige tekens onverminkt worden weggeschreven naar andere geheugenplaatsen c.q. externe geheugens?

Ten einde deze vragen te kunnen beantwoorden zullen we moeten vaststellen:

1. Hoe de gegevens in de computer worden gerepresenteerd: de tekencode.
2. Is de code foutontdekkend, bijvoorbeeld door middel van paritycheck?
3. Is de code wellicht foutherstellend (Hammingcode)?
4. Vindt er read-after-write plaats?
5. Hoe worden korte stroomstoringen opgevangen?

De vraag is nu, hoe bepaalde tekortkomingen kunnen worden ondervangen. Het verdient mijns inziens aanbeveling om waar dit mogelijk is gebruik te maken van controletellingen, voornummering van formulieren

c.q. batches en andere algemeen geldende inputcontrolemaatregelen. zoals geprogrammeerde validatiecontroles, record-editing met behulp van vaste gegevens en dergelijke. De eis is dat het optreden van een fout onderkend kan worden.
Wanneer er een fout wordt gesignaleerd en de produktie zonder bezwaar kan worden herhaald, kunnen we desondanks spreken van voldoende beveiliging.
Anders is het wanneer geen reproduktie mogelijk is, zoals bij interactieve update van stambestanden waarbij de oorspronkelijke gegevens overschreven worden. Dan is het noodzakelijk back-ups aan te houden waarmee eventueel de verwerking herhaald kan worden.
Wenselijk is het bovendien de invoergegevens op een machinaal leesbaar medium vast te leggen (log-file).
Daarenboven dient de organisatie waarborgen te bieden tegen het ongeautoriseerd inbrengen of lezen van gegevens in bestanden. Soms kan hierbij de besturingsprogrammatuur behulpzaam zijn, doordat een speciale autorisatiecode wordt verlangd. Deze code dient uitsluitend bekend te zijn bij voor invoer geautoriseerde functionarissen.

IV Verwerking van de gegevens

Zoveel als mogelijk dient ook hier gebruik te worden gemaakt van controletellingen en verbandscontroles. Meer nog dan bij de invoer zijn hier de specifieke eigenschappen van de machine van belang.
Een aantal hiervan zijn:

a. De wijze waarop het geheugen is geconstrueerd: blijven gegevens na een stroomonderbreking bewaard? Zo ja, dan kan de verwerking ongestoord worden voortgezet na het weer aanzetten van de machine; bij het starten van een nieuw programma dient dan echter het geheugen "schoon"gemaakt te worden. Zo nee, dan zijn tijdens de verwerking van eigszins omvangrijke jobs "save- en restore-procedures" noodzakelijk.

b. Wanneer geen foutontdekkende code wordt gebruikt is het risico aanwezig dat de verminking van een teken resulteert in een (andere) besturingsopdracht. Met de leverancier moet worden nagegaan hoe hiertegen moet worden opgetreden.

c. Van belang kan zijn de gebruikte programmeertaal. Sommige talen staan toe dat de verwerking wordt gestaakt, het programma vervolgens gemodificeerd, geheugenvelden worden gewijzigd en dergelijke, waarna de verwerking kan worden voortgezet. (Bijvoorbeeld: Basic, Business-basic.) N.B.: Extended-basic en thans ook het MAI-business-basic hebben echter de mogelijkheid om in het programma instructies op te nemen die onderbreking van dat programma onmogelijk maken.

d. De wijze waarop getallen in de computer worden gerepresenteerd. Zo zijn er operating systems waarbij alleen met getallen kan worden gerekend als deze in de "floating-point"-representatie staan, terwijl tevens slechts 6 of 8 cijfers nauwkeurig kunnen worden

weergegeven. Er kunnen al vrij snel onaanvaardbare afrondingsverschillen optreden die in administratieve toepassingen ongewenst zijn. Er zal dan na iedere bewerking een afronding moeten worden geprogrammeerd.

e. Is er een "harde" beveiliging tegen het overschrijven van bestanden? Vooral de cassette of cartridge is niet vrij van risico's!

V Uitvoer

Wederom is controle door middel van tellingen het meest aan te bevelen. Verficatie zal veelal handmatig moeten plaatsvinden (verband met invoer).

VI Enige suggesties

Algemeen kunnen nog enkele suggesties onder de aandacht worden gebracht:

a. Wanneer niet volgens een strak schema wordt gewerkt, dient de voortgang van de verwerking in een controleregister te worden vastgelegd. Dit voorkomt ontijdige, dus onjuiste updates, het overslaan van verwerkingen en dergelijke.
Een en ander dient te worden gezien in het kader van een aanvulling op de meestal zwakkere interne controle, zoals ontbreken van voldoende functiescheiding, gemakkelijke toegang tot de apparatuur en dergelijke.

b. Wanneer meerdere beeldbuisstations zijn aangesloten, dienen deze onderling te zijn afgeschermd als dit vanuit de toepassingen wordt vereist. Per toepassing dient de autorisatie geregeld te zijn.

c. Voorzie zoveel mogelijk in duplicering van belangrijke bestanden.

d. Beveilig magneetbandcassettes en -cartridges tegen ontijdige overschrijving. Zo mogelijk dienen ook andere informatiedragers beschermd te worden, anders dienen geprogrammeerde oplossingen te worden overwogen.

e. Ga na of tape-labelling wordt toegepast en zo ja, in hoeverre deze voor beveiligingsdoeleinden dienstig is.

f. Kan worden vastgesteld met welk programma c.q. welke versie daarvan de verwerking heeft plaatsgevonden? Zo niet, dan dienen compilaties in een register te worden ingeschreven.

g. Test iedere nieuwe toepassing, alsmede wijzigingen in bestaande toepassingen met behulp van een representatieve testset.

h. Ga na of door het besturingssysteem een logfile wordt opgebouwd en welke gegevens daarin worden opgenomen.

VII Tenslotte

Dit artikel is slechts een schoorvoetend betreden van het terrein van de beveiliging van geautomatiseerde administratieve gegevensverwerking in kleinere organisaties.
Suggesties van lezers, onder meer ter zake van de gegeven opsommingen, zijn zeer welkom en kunnen dienen ter uitbreiding en verbetering van het onderzoek op dit gebied.
Ook problemen die in de controle naar voren komen en aan ons ter kennis worden gebracht, kunnen het beeld completeren en wellicht leiden tot een lijst met attentiepunten bij verwerking op kantoorcomputers.

COMPACT is een uitgave van de AC-groep van Klynveld Kraayenhof & co.

DE ZEEFMETHODE ALS SELECTIEMETHODE VOOR STATISTISCHE STEEKPROEVEN IN DE CONTROLEPRAKTIJK (IV)

door C. Rietveld

Inleiding

In mijn vorig artikel heb ik in hoofdstuk II de toepassing van zeefgrenzen als selectiecriterium toegelicht'). Hierna zal blijken, dat het nut van zeefgrenzen vooral bij selectiecombinaties evident is. De in mijn artikel gehanteerde aanduidingen worden opnieuw binnen een kader weergegeven.

T = totaalbedrag der verantwoording
m = vooraf bepaalde steekproefomvang
n = aantal in feite geselecteerde, te controleren posten
p = bovengrens van de fractie fouten in de verantwoording
P = bepaalde post, alsmede bedrag van die post
F = grootte van fout in post P
M = zeefmaximum voor een steekproefomvang m (T/m)
a = a-select getal voor post P
Z = zeefgetal voor post P en een steekproefomvang m (M * a)
X = zeefgrensmaximum voor post P (T/P)
G = zeefgrens voor post P (X * a)

Voor een goed begrip van de selectiecombinaties wil ik vooraf een beknopt overzicht geven van de relatie tussen steekproefomvang, zeefgetallen en zeefgrenzen. Daarbij is een en ander ook algebraïsch tot uitdrukking gebracht, echter uitsluitend voor degenen die daarin steun vinden voor een goed begrip van het overzicht.

') In bedoeld artikel dienen de navolgende verbeteringen te worden aangebracht:

Pag. 4: (M $\leqslant$ 359) moet zijn (m $\leqslant$ 359)
Pag. 5: (zie tabel op pag. 4) moet zijn (zie tabel op pag. 3)
Pag. 5: laatste regel: zeefgrenzen moet zijn zeefgetallen
Pag. 6: paragraaf 3 moet zijn paragraaf 4
Pag. 7: (zie pag. 4) moet zijn (zie pag. 5)

Het zeefmaximum is:
- verantwoordingstotaal/steekproefomvang. $M = T/m$

Dit houdt in, dat een hogere (lagere) steekproefomvang leidt tot een lager (hoger) zeefmaximum.

Het zeefgetal is:

- zeefmaximum * a-select getal (<1). $Z = M * a = T/m * a$

Een hogere (lagere) steekproefomvang leidt derhalve ook tot een lager (hoger) zeefgetal.

Dit kenmerk bepaalt mede het grote toepasbaarheidsbereik en de grote efficiency van de zeefmethode.

Selectie vindt plaats als de vraag:

- post > zeefgetal? $P > T/m * a?$

met "ja" wordt beantwoord.

Niet geselecteerd worden derhalve een:

- post < zeefgetal $P < T/m * a$
- post = zeefgetal $P = T/m * a$

Dit laatste zeefgetal geeft de kritische grens aan, te weten het zeefgetal waarbij de post nog juist niet zou worden geselecteerd.

De steekproefomvang, waarbij de kritische grens wordt bereikt heet de zeefgrens, dit is de steekproefomvang waarbij de post nog juist niet zou worden geselecteerd.

Als $P = T/m * a$
is $G = m = T/p * a$

Derhalve:

- Bij een steekproefomvang = zeefgrens
  is het zeefgetal = post

Als $m = G$
is $Z = P$

Gelet op de relatie tussen steekproefomvang en zeefgetal dus ook:

- Is steekproefomvang > zeefgrens
  dan is zeefgetal < post
  ofwel post > zeefgetal
  zodat post wordt geselecteerd

Als $m > G$
is $Z < P$
$P > Z$

Als de vraag:

- post > zeefgetal? $P > Z?$

met "ja" wordt beantwoord (selectie), geldt dit derhalve ook voor de alternatieve vraag: $P > T/m * a?$ $m > T/P * a?$

- steekproefomvang > zeefgrens? $m > G?$

De zeefgrens kan als volgt worden berekend.
Allereerst berekenen we het zeefgrensmaximum:

- verantwoordingstotaal/post $X = T/P$

Vervolgens de zeefgrens:

- zeefgrensmaximum * a-select getal $G = T/P * a$

Uit het voorgaande moge blijken, dat zeefgetallen en zeefgrenzen als selectiecriteria mathematisch een gelijke betekenis hebben.

IV SELECTIECOMBINATIES

1. Wat zijn selectiecombinaties?

Een verantwoording of deel van een verantwoording kan aan meedere steekproeven onderworpen worden. Zo kan bijvoorbeeld een inkoopregistratie gecontroleerd worden op overeenstemming met respectievelijk: inkoopfactuur, ontvangstenregistratie en orderbescheiden, waarbij voor elke controlehandeling de steekproefomvang verschillend is en onder meer de uitkomsten van de steekproeven afzonderlijk worden geëvalueerd. Er kan van een selectiecombinatie worden gesproken, als de selecties voor alle steekproeven gecombineerd in één arbeidsgang worden verricht, waarbij zoveel mogelijk dezelfde posten worden geselecteerd, dat wil zeggen een uiterste minimalisering van het aantal te controleren posten wordt verkregen. Bij een steekproefomvang van respectievelijk 500, 280 en 120 zou het aantal te controleren posten bij gescheiden selecties 900 (500 + 280 + 120) zijn; bij een selectiecombinatie wordt dit aantal teruggebracht tot 500 posten, waarvan de 280 en de 120 posten deel uitmaken.

Deze efficiënte trekkingstechniek is voor zover ik heb kunnen nagaan alleen bij de zeefmethode mogelijk. Hiertoe zijn geen bijzondere maatregelen vereist, behoudens de uitnutting van een hiervoor aangeduid kenmerk van de zeefmethode, te weten de directe relatie tussen steekproefomvang en zeefgetal.

Allereerst wordt de procedure beschreven op basis van zeefgetallen, vervolgens die op basis van zeefgrenzen.

2. Selectiecombinaties op basis van zeefgetallen

Nemen we aan dat vorenvermelde inkoopverantwoording een verantwoordingstotaal heeft van 11,5 mln. en 60.000 posten omvat.
De controle richt zich op de overeenstemming van:

a. geboekte bedragen met intern gefiatteerde en gecontroleerde facturen;
b. facturen met geregistreerde ontvangsten;
c. facturen met orderbescheiden.

Het steekproefplan richt zich op de uitspraak met 99% betrouwbaarheid ten aanzien van elke controlehandeling afzonderlijk, dat de fouten in totaal - na afronding - ten hoogste respectievelijk f 120.000, f 200.000 en f 350.000 zullen bedragen. Daarnaast wordt een gecombineerde uitspraak verlangd. In de verwachting dat in de steekproef geen fouten zullen voorkomen, wordt de steekproefomvang (m) gesteld op respectievelijk 440, 265 en 150. Deze aantallen worden ontleend aan de tabel op basis van de Poisson-verdeling, zoals deze in Hoofdstuk I beknopt werd opgenomen.
De berekening van deze aantallen is als volgt:

$$4{,}6 \text{ (vlg. tabel)} = m * \frac{120.000 \text{ resp. } 200.000 \text{ resp. } 350.000}{11{,}5 \text{ mln.}}$$

De zeefmaxima voor a, b en c zijn respectievelijk 26.136, 43.396 en 76.666, te weten 11,5 mln. gedeeld door respectievelijk 440, 265 en 150.

Voor een nader uitgewerkt voorbeeld van de selectieprocedure ga ik uit van de posten, die eerder ten tonele werden gevoerd.

Nrs.	A-select getal	Posten	Zeefgetallen			Te controleren?		
			a	b	c	Post > zeefgetal?		
	(a)	(P)	(Z_a)	(Z_b)	(Z_c)	(Z_a?)	(Z_b?)	(Z_c?)
1	0,22683	3.780	5.928	(9.843)	(17.390)	N	(N)	(N)
2	0,66041	14.720	17.260	(28.659)	(50.630)	N	(N)	(N)
3	0,00846	1.150	221	367	648	J	J	J
4	0,65429	7.715	17.100	(28.393)	(50.161)	N	(N)	(N)
5	0,08035	2.570	2.100	3.486	(6.160)	J	N	(N)
6	0,77440	56.230	20.239	33.605	59.370	J	J	N

J = ja; N = neen; ( ) = blijft bij beperkte procedure achterwege.

We volgen de selectieprocedure voor post 1 op de voet. In deze procedure wordt voor alle steekproeven per post hetzelfde a-selecte getal gebruikt.

a) Bereken zeefgetal voor steekproef a (Z_a):
   - zeefmaximum (M_a) * a-select getal (a)

     26.136 * 0,22683 = 5.928 (Z_a)

   Toets:
   - post (P) > zeefgetal voor a (Z_a)?
     3.780 > 5.928?

   Conclusie:
   - neen, dus geen selectie voor steekproef a.

b) Bereken zeefgetal voor steekproef b (Z_b):

- zeefmaximum (M_b) * a-select getal (a)

  43.396 * 0,22683 = 9.843 (Z_b)

Toets:
- post (P) > zeefgetal voor b (Z_b)?
  3.780 > 9.843?

Conclusie:
- neen, dus geen selectie voor steekproef b.

c) Bereken zeefgetal voor steekproef c (Z_c):

- zeefmaximum (M_c) * a-select getal (a)

  76.666 * 0,22683 = 17.390 (Z_c)

Toets:
- post (P) > zeefgetal voor c (Z_c)?
  3.780 > 17.390?

Conclusie:
- neen, dus geen selectie voor steekproef c.

De steekproeven zijn in de procedure bewust in afnemende volgorde van de steekproefomvang (440, 265 en 150) opgenomen. Dit leidt derhalve - bij de hantering van hetzelfde a-selecte getal - tot een toenemende volgorde van de zeefgetallen: 5.928, 9.843 en 17.390.

Aangezien blijkt:

- post < zeefgetal voor a (Z_a)
  3.780 < 5.928

geldt dit ook voor de hogere zeefgetallen voor b en c (9.843 en 17.390). Een "neen"-conclusie voor een steekproef, geldt zonder nadere berekening ook voor de volgende steekproeven. Dit impliceert dat voor post 1 de procedure tot steekproef a kan worden beperkt.

Deze beperking is kwantitatief belangrijk. Deze geldt immers in ons voorbeeld de 59.560 niet-geselecteerde posten. Alleen voor de 440 voor steekproef a geselecteerde posten dient de procedure te worden uitgebreid om vast te stellen in hoeverre zij ook voor de steekproeven b en c worden geselecteerd.

Hieruit vloeit voort, dat de volgende selectieconclusies kunnen voorkomen. Tevens worden de aantallen posten vermeld, die bij benadering op basis van die conclusies worden geselecteerd.

Aantallen				Selectieconclusies		
Geselecteerd			Niet geselecteerd	a	b	c
a	b	c		(440)	(265)	(150)
-	-	-	59.560	N	(N)	(N)
175	-	-		J	N	(N)
115	115	-		J	J	N
150	150	150		J	J	J
440	265	150	59.560			

Aangezien een "neen"-conclusie altijd door een "neen"-conclusie wordt gevolgd, wordt, zoals ook uit het schema blijkt, een "ja"-conclusie altijd door een "ja"-conclusie voorafgegaan. Een selectie voor steekproef c respectievelijk b impliceert derhalve een selectie voor steekproef b respectievelijk a; met andere woorden de 150 voor c geselecteerde posten zijn begrepen in de 265 voor b geselecteerde posten, en deze weer in de 440 voor a geselecteerde posten.

Door deze systematiek wordt derhalve zowel een efficiënte selectieprocedure als een minimalisering van het aantal te controleren posten tot stand gebracht.

Overigens zij erop gewezen, dat post 6 groter is dan de zeefmaxima voor steekproeven a en b, te weten 26.136 en 43.396 en derhalve een 100% trefkans hebben. Een fout in deze post leidt tot een separate behandeling bij de evaluatie, zoals eerder besproken.

Naast de geselecteerde posten dienen de zeefgetallen voor de steekproeven met een "ja"-conclusie te worden vastgelegd.
Geconstateerde fouten die blijkens de vastlegging kleiner zijn dan de zeefgetallen kunnen immers voor de evaluatie buiten beschouwing blijven.

Voor de afzonderlijke evaluatie van de steekproefuitkomsten van de steekproeven a, b en c zij verwezen naar Hoofdstuk I. Op de bijzondere problematiek van een gecombineerde uitspraak op grond van gezamenlijke steekproefuitkomsten wordt op deze plaats niet ingegaan omdat dit ons te ver van het eigenlijke onderwerp voert.

De zeefmethode biedt derhalve bij selectiecombinaties de volgende faciliteiten:

- uitvoering van de selecties in één arbeidsgang;
- beperking van de selectieprocedure voor niet-geselecteerde posten;
- automatische minimalisering van het aantal te controleren posten.

Een verdere vereenvoudiging in de procedure en in de vastlegging van de geselecteerde posten wordt gerealiseerd door de gebruikmaking van zeefgrenzen in plaats van zeefgetallen.

3. Selectiecombinaties met zeefgrenzen

De procedure op basis van zeefgrenzen wordt geïllustreerd aan de hand van het in paragraaf 2 besproken voorbeeld.
Met zeefgrenzen wordt de tabel als volgt:

Nrs.	A-select getal	Posten	Zeefgrens	Te controleren? Omvang > zeefgrens?		
	(a)	(P)	(G)	440 > G?	265 > G?	150 > G?
1	0,22683	3.780	690	N	(N)	(N)
2	0,66041	14.720	515	N	(N)	(N)
3	0,00846	1.150	84	J	J	J
4	0,65429	7.715	975	N	(N)	(N)
5	0,08035	2.570	359	J	N	(N)
6	0,77440	56.230	158	J	J	N

J = ja; N = neen; ( ) = blijft bij beperkte procedure achterwege.

Ook hier volgen we eerst de selectieprocedure voor post 1.

a) Bereken zeefgrensmaximum (X):
- verantwoordingstotaal (T) / post (P)
  11,5 mln./3.780 = 3.042 (X)

Bereken zeefgrens (G):
- zeefgrensmaximum (X) * a-select getal (a)
  3.042 * 0,22683 = 690 (G)

Toets voor steekproef a:
- steekproefomvang (m_a) > zeefgrens (G)?
  440 > 690?

Conclusie:
- neen, dus geen selectie voor steekproef a.

b) Toets voor steekproef b:
- steekproefomvang (m_b) > zeefgrens (G)?
  265 > 690?

Conclusie:
- neen, dus geen selectie voor steekproef b.

c) Toets voor steekproef c:
- steekproefomvang (m_c) > zeefgrens (G)?
  150 > 690?

Conclusie:
- neen, dus geen selectie voor steekproef c.

Zowel de tabel als de beschrijving van de procedure illustreren de vereenvoudiging. In plaats van de berekening van en toetsing aan drie zeefgetallen is er de berekening van één zeefgrens en drie toetsingen met eenzelfde zeefgrens.

Ook hier blijkt de procedure beperkt te kunnen worden. Een "neen"-conclusie voor steekproef a op grond van een steekproefomvang $\leqslant$ zeefgrens impliceert ook een "neen" voor de volgende steekproeven met een lagere steekproefomvang, zodat de procedure voor post 1 tot steekproef a wordt beperkt.

Het eerder besproken schema met selectieconclusie is ook hier van toepassing.

In plaats van de vastlegging van de drie zeefgetallen bij de geselecteerde posten komt de vastlegging van één zeefgrens. Deze zeefgrens kan bij de evaluatie zonodig tot zeefgetallen worden herleid.

Een nadeel is dat zeefgrenzen worden berekend op basis van een per post te berekenen zeefgrensmaximum. Dit geeft derhalve meer rekenwerk dan de berekening van zeefgetallen op basis van een voor alle posten vastgesteld zeefmaximum.

Een optimaal resultaat wordt daarom bereikt door:

- primaire selectie op basis van de laagste zeefgetallen (steekproef a);
- na selectie op grond van "ja"-conclusie: berekening van de zeefgrens;
- voortzetting van de procedure met deze zeefgrens.

Het gebruik van zeefgrenzen voor tussentijdse uitbreidingen en beperkingen in de steekproefomvang (zie hoofdstuk II) is ook voor selectiecombinaties mogelijk en wel voor één of meer van de gecombineerde steekproeven in een mate, die per steekproef kan worden bepaald. Ten aanzien van uitbreidingen wordt deze faciliteit uiteraard begrensd door de omvang van de vooraf extra geselecteerde posten.

4. Combinatiemogelijkheden

Het zou te ver voeren op alle combinatiemogelijkheden in te gaan en ik wil mij daarom tot een aantal voorbeelden beperken.

Het in paragraaf 2 en 3 toegelichte combinatiemodel is zonder meer te gebruiken voor het geval dat een verantwoording door twee instanties, bijvoorbeeld de interne controle-afdeling en de externe accountant, wordt onderzocht ter verkrijging van een zelfstandig oordeel over de verantwoording. Het wordt daarbij doelmatig geoordeeld, dat de geselecteerde posten voor de externe accountant deel uitmaken van die voor de interne controle-afdeling. Bij toepassing van het besproken model komt in plaats van de splitsing naar controlehandelingen, die naar controlerende instanties.

Indien het echter de wens zou zijn de posten voor de interne controle-afdeling en die voor de externe accountant zo min mogelijk te doen samenvallen, dient per post gebruik te worden gemaakt van twee a-selecte getallen getrokken uit twee afzonderlijke reeksen.

Een andersoortige combinatie betreft een verantwooring, samengesteld uit een aantal deelverantwoordingen, waarbij de totale verantwoording en de afzonderlijke deelverantwoordingen aan separate steekproeven worden onderworpen.
Men denke bijvoorbeeld aan een holding met een aantal dochterondernemingen, die bijvoorbeeld het schadeverzekeringsbedrijf uitoefenen. De accountant onderzoekt de schade-uitkeringen steekproefsgewijs en verlangt daartoe op basis van de uitkomsten van die steekproef afzonderlijke uitspraken per dochteronderneming en voor de gezamenlijke dochterondernemingen.
Deze eis hangt samen met de afgifte van accountantsverklaringen voor de geconsolideerde jaarrekening en voor de jaarrekening van elke dochteronderneming afzonderlijk.
Per deelverantwoording omvat de combinatie de steekproef voor de gezamenlijke vennootschappen en die voor de betrokken dochtervennootschap.
Per post zijn er twee zeefgrenzen, te weten die voor de totaalverantwoording en die voor de verantwoording van de betrokken vennootschap, gebaseerd op hetzelfde a-selecte getal:

- totaaluitkeringen van alle vennootschappen/post * a-select getal;
- totaaluitkeringen van de betrokken vennootschap/post * a-select getal.

De procedure verloopt overigens in principe als onder 2 en 3 beschreven met een primaire toetsing op basis van het laagste zeefgetal.

Een verdere complicatie is er wanneer wij in vorenvermeld voorbeeld veronderstellen dat er drie schadecategorieën zijn, die voor de vennootschappen te zamen in drie afzonderlijke schade-afdelingen behandeld worden. Men wil ter beoordeling van de kwaliteit van de drie afdelingen steekproeven gericht op de uitkeringen per afdeling met een voor dat doel vastgestelde steekproefomvang per afdeling. De verantwoording per vennootschap valt dan uiteen in drie deelverantwoordingen die elk in drie steekproeven zijn betrokken (per vennootschap, per afdeling, totaal-generaal).
Per post zijn er drie verschillende zeefgetallen.

Dankzij de zeefmethode is dit ogenschijnlijk gecompliceerde probleem simpel op te lossen, ook hier resulterend in:

- een gecombineerde selectie in <u>één arbeidsgang</u>;
- <u>beperking</u> van de <u>selectieprocedure</u> voor niet-geselecteerde posten;
- een <u>minimaal</u> aantal te controleren posten;
- de mogelijkheid tot tussentijdse <u>uitbreiding</u> en <u>beperking</u> van de omvang per steekproef.

Ook een steekproefsgewijze controle op herverzekeringen kan op efficiënte wijze in de selectieprocedure worden ingebouwd.

Zonder op deze plaats in details te treden, wil ik erop wijzen, dat steekproeven op de goederenbeweging een uitgebreide mogelijkheid bieden voor selectiecombinaties. Uitgaande van de besproken principes voor selectiecombinaties zijn ter oplossing van de specifieke problematiek en ter bevordering van de efficiency een aantal op de controle van de goederenbeweging gerichte technieken ontwikkeld.

Uit mijn uiteenzettingen over de zeefmethode in drie artikelen van Compact moge U met mij de conclusie trekken, dat deze methode in zijn huidige vorm zich kenmerkt door een grote mate van flexibiliteit, efficiency en toepasbaarheid. Dit geldt voor toepassing bij steekproeven in het algemeen, maar voor combinaties van steekproeven in het bijzonder.
Daarbij is te denken aan de volende aspecten:

- eenvoudige mogelijkheid tussentijds de steekproefomvang aan gewijzigde omstandigheden aan te passen;
- flexibiliteit bij de aanpassing aan de behoeften van de gebruiker;
- beperking van de invoer van gegevens, voor zover deze invoer handmatig geschiedt (onder meer subselectie-techniek, beschreven in hoofdstuk III);
- efficiënte selectieprocedures met name bij selectiecombinaties;
- minimalisering van te controleren posten bij selectiecombinaties.

Ik vertrouw dat mijn artikelen U inzicht hebben gegeven in de verwerking van de zeefmethode en U overtuigd hebben van het nut van de toepassing van die methode. Mochten er hier en daar bij U nog vragen zijn opgekomen, dan ben ik gaarne bereid U schriftelijk of mondeling nadere toelichtingen te geven.

COMPACT is een uitgave van de AC-groep van Klynveld Kraayenhof & co

COMBI (CObol Missed Branch Indicator)

door A. van der Drift

Inleiding

Combi is een pakket, dat als hulpmiddel dient bij het testen van computerprogramma's die geschreven zijn in de programmeertaal COBOL.
Het pakket is in 1969 ontwikkeld door Klynveld Kraayenhof & co, waarna een softwarehouse een eigen versie van dit pakket enige tijd op de markt bracht. Thans levert KKC het pakket, dat voor bijna alle versies van COBOL op alle computersystemen beschikbaar is, na eventueel een geringe aanpassing te hebben ondergaan.

Testen van programmatuur

De standaardmethode voor het testen van een programma is het gebruik van testinvoer, dat aan de hand van de gebruikerseisen, zoals deze zijn vastgelegd in de systeem- en programmabeschrijving, wordt vervaardigd.
Vooraf dient bepaald te worden hoe het programma de testinvoer behoort te verwerken; met andere woorden, wat de resultaten van de verwerking dienen te zijn. Bij vergelijking met de uitvoer van het programma wordt geconstateerd of de verwerking van de testinvoer juist is geschied.
Indien fouten in de verwerking worden geconstateerd, dient het programma verbeterd en de test herhaald te worden tot de juiste resultaten worden verkregen.
Na afloop van een test, waarvan de uitkomsten overeenstemmen met de voorberekende resultaten, kan toch niet met zekerheid worden gesteld dat het programma volledig voldoet aan zijn beschrijving. Het is immers niet zeker dat alle instructies zijn doorlopen tijdens de verwerking van de testinvoer.
Het programma kan nog instructies bevatten, die nooit of slechts bij verwerking van andere dan de gebruikte testinvoer worden uitgevoerd. De testinvoer kan derhalve incompleet zijn. Het is veelal niet mogelijk voor grote, complexe programma's complete testinvoer te vervaardigen.
Anderzijds kan de testinvoer schijnbaar onvolledig zijn, doordat het programma instructies bevat, die niet voldoen aan de programmaspecificaties.

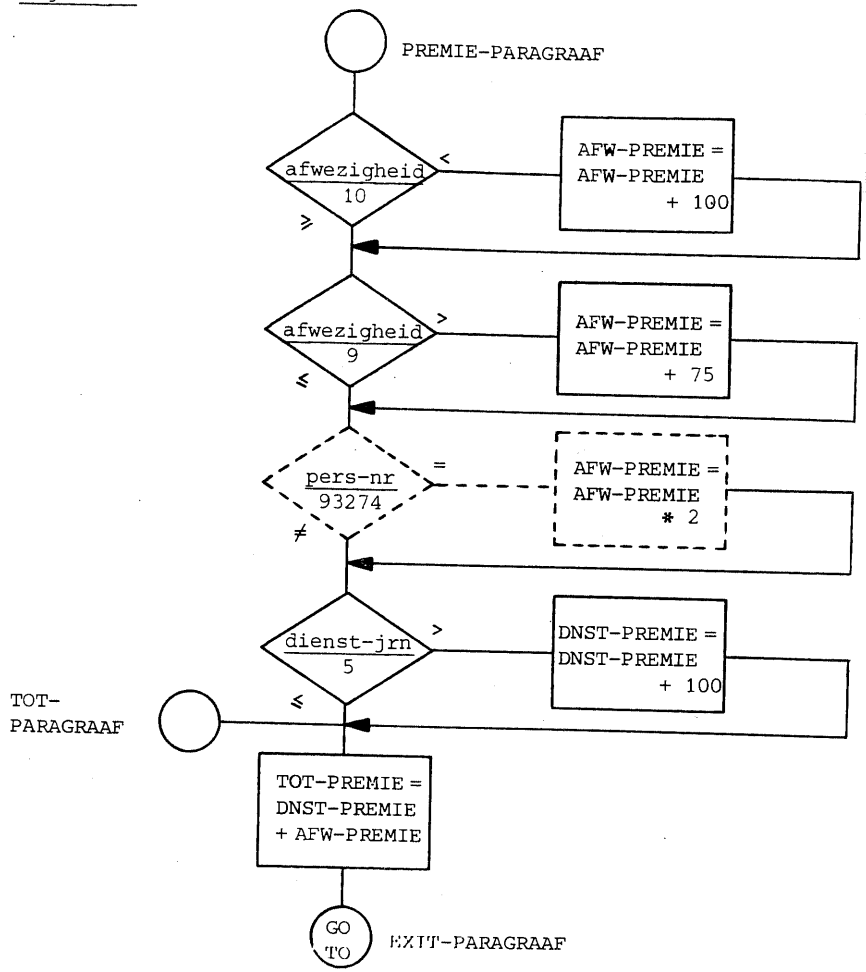
Ter illustratie zijn een aantal delen van een programma opgenomen, die op basis van bepaalde criteria premieberekeningen uitvoeren.

Figuur 1a toont een onderdeel van een programma, waarin opgenomen de aanwezigheid van een instructie, die in een specifieke situatie zal worden uitgevoerd. Figuur 1b geeft hetzelfde deel van het programma in blokschema weer.
De testinvoer zal naar alle waarschijnlijkheid geen post bevatten, die aan conditie 3 voldoet, waardoor bij het testen van het programma instructie 3 niet zal worden uitgevoerd. De testuitvoer zal in dit geval dan juist zijn, omdat instructie 3 niet mag worden uitgevoerd, daar deze niet staat beschreven in de programmabeschrijving.

Figuur 1a

```
PREMIE-PARAGRAAF.

    IF  AFWEZIGHEID < 10                          conditie 1
        ADD 100 TO AFW-PREMIE.                    instructie 1

    IF  AFWEZIGHEID > 9                           conditie 2
        ADD 75 TO AFW-PREMIE.                     instructie 2

    IF  PERS-NR = 93274                           conditie 3
        MULTIPLY AFW-PREMIE BY 2.                 instructie 3

    IF  DNST-JRN > 5                              conditie 4
        ADD 100 TO DNST-PREMIE.                   instructie 4

TOT-PARAGRAAF.

    ADD DNST-PREMIE, AFW-PREMIE TO TOT-PREMIE.    instructie 5

    GO TO EXIT-PARAGRAAF.                         instructie 6
```

Figuur 1b



Figuur 2 bevat een gedeelte uit hetzelfde COBOL-programma, waarbij een bepaalde instructie (instructie 5) nooit zal worden uitgevoerd en ook nooit zal mogen worden uitgevoerd. Bij het testen van het programma zal deze instructie dan ook geen foutief resultaat teweeg brengen, maar onnodig beslag leggen op geheugenruimte van de computer.

Figuur 2

```
PREMIE-PARAGRAAF.
    IF  AFWEZIGHEID < 10                    conditie 1
        ADD 100 TO AFW-PREMIE               instructie 1
        GO TO TOT-PARAGRAAF.                instructie 2
    IF  AFWEZIGHEID > 9                     conditie 2
        ADD 75 TO AFW-PREMIE                instructie 3
        GO TO TOT-PARAGRAAF.                instructie 4
    ADD AFW-PREMIE TO TOT-PREMIE.           instructie 5
TOT-PARAGRAAF.
    ADD AFW-PREMIE TO TOT-PREMIE .          instructie 6
    GO TO EXIT-PARAGRAAF.                   instructie 7
```

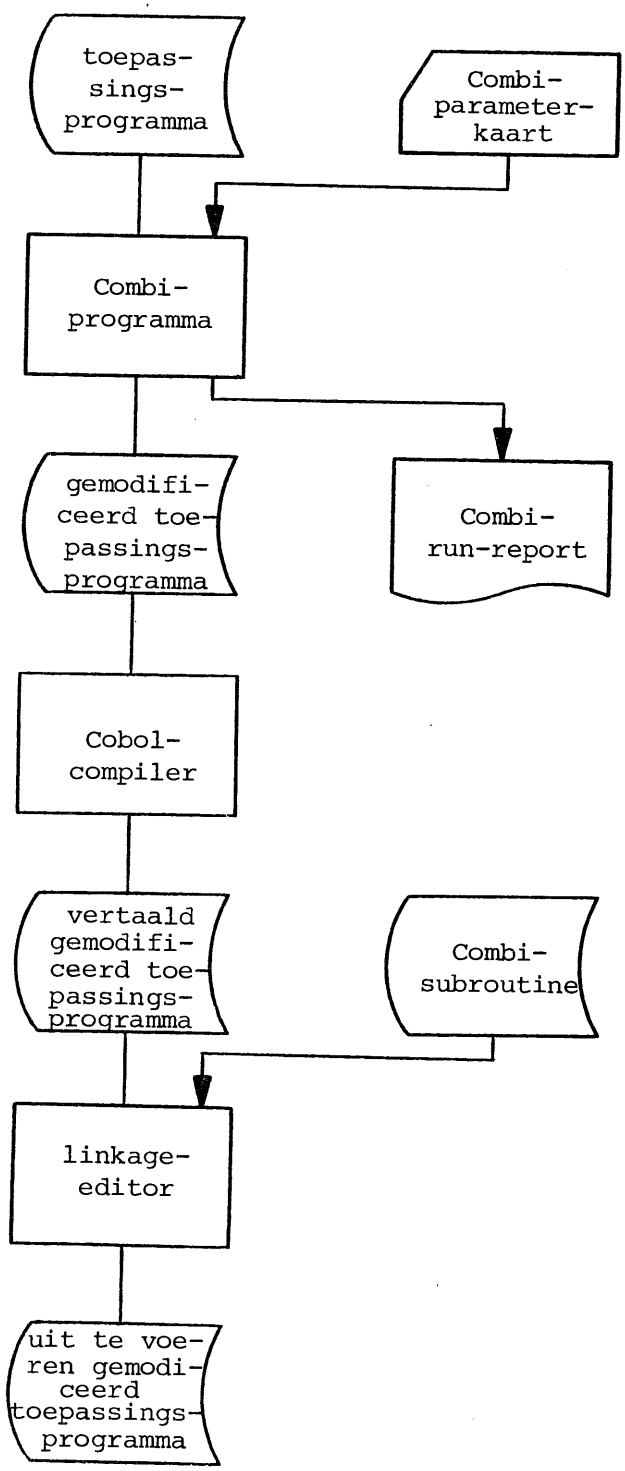
Het beoordelen van een programma op grond van de programmalijst, die bestaat uit bestands-, record- en velddefinities en instructies, waarmee de invoer veelal afhankelijk van condities verwerkt wordt (zogenaamde source code-review) zal bij grote, complexe programma's veelal geen uitkomst kunnen bieden voor het ontdekken van de in de figuren 1 en 2 geschetste foutsituaties. Het laat zich raden, dat zich zeker complexere (minder rechtlijnige) foutsituaties kunnen voordoen, dan die hiervoor werden geïllustreerd.

Combi-pakket

Zoals uit het voorgaande duidelijk zal zijn, is het van belang te weten, welke instructies niet zijn uitgevoerd gedurende de testverwerking, waardoor een beter oordeel gevormd kan worden over de werking van het programma en de testinvoer.
Combi verschaft de mogelijkheid inzicht te verkrijgen in de niet uitgevoerde instructies bij (test)uitvoering van een COBOL-programma.
Combi kan tevens bepalen hoeveel keer de wel uitgevoerde instructies gedurende een bepaalde verwerking zijn doorlopen. Hiermee kan inzicht verkregen worden in de mate van efficiency van verwerking.
Het Combi-pakket bestaat uit een programma en een subroutine (deelprogramma), beide geschreven in ANS-COBOL.

Het Combi-programma modificeert het te testen toepassingsprogramma. Na modificatie dient het toepassingsprogramma voor uitvoering vertaald te worden (gecompileerd en gelinkt), waarbij de Combi-subroutine aan het toepassingsprogramma dient te worden gekoppeld. De sprong vanuit het gemodificeerd toepassingsprogramma naar de subroutine, wordt veroorzaakt door de aangebracht modificaties (zie figuur 3).

Figuur 3



Werking

De modificaties bestaan uit:

- het toevoegen van hulpvelden en instructies, ten einde deze velden te initialiseren,
- het tussenvoegen van speciale instructies, en
- het toevoegen van de spronginstructie naar de Combi-subroutine.

De door Combi aangebracht modificaties tasten de werking van het toepassingsprogramma niet aan, zodat een zelfde uitvoer wordt verkregen als bij uitvoering van het niet gemodificeerde programma. De modificaties vergroten het programma echter wel. De mate van vergroting hangt af van de grootte van het programma.
Gedurende de uitvoering van het gemodificeerde toepassingsprogramma zorgen de tussengevoegde speciale Combi-instructies voor tellingen, ten einde te bepalen hoeveel keer de instructies van het toepassingsprogramma zijn uitgevoerd. De Combi-instructies gebruiken de aan het toepassingsprogramma toegevoegde hulpvelden voor het opslaan van de tellingen.
Na beëindiging van de verwerking van het toepassingsprogramma zorgt de toegevoegde spronginstructie voor het uitvoeren van de aangekoppelde Combi-subroutine. Deze subroutine drukt de inhoud van de Combi-hulpvelden af. Hiermee wordt een lijst verkregen met de niet uitgevoerde instructies of, indien gewenst, het aantal keer dat instructies zijn uitgevoerd.

Combi bepaalt ten gevolge van zijn werking niet altijd per instructie de hoeveelheid uitvoering, waarvan bij figuur 4 een toelichting wordt gegeven.

Figuur 4

```
PRINT-PARAGRAAF.
      MOVE AFW-PREMIE TO AFW-PRINT.        instructie 1
      MOVE DNST-PREMIE TO DNST-PRINT.          "      2
      MOVE PERS-NR TO PERS-PRINT.              "      3
      WRITE REGEL.                             "      4
      MOVE SPACES TO REGEL.                    "      5
      GO TO EXIT-PARAGRAAF.                    "      6
```

Indien instructie 1 tienmaal wordt uitgevoerd, worden de instructies 2 tot en met 6 eveneens tienmaal uitgevoerd. Derhalve is het voor Combi niet noodzakelijk om per instructie het aantal uitvoeringen te bepalen. Combi beperkt het toevoegen van instructies daarmee tot het noodzakelijke.

Figuur 5

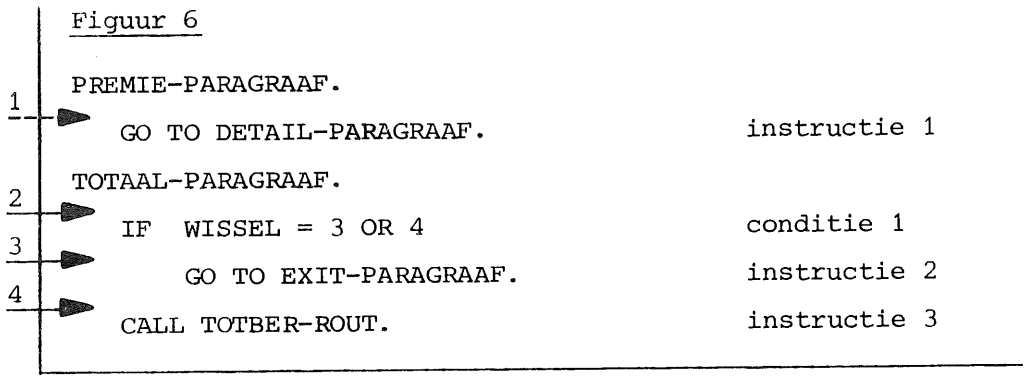
```
1 PREMIE-PARAGRAAF.
2     IF  AFWEZIGHEID < 10                  conditie 1
          ADD 100 TO AFW-PREMIE             instructie 1
      ELSE
3         GO TO EXIT-PARAGRAAF.             instructie 2
      IF  DIENST-JRN > 5                    conditie 2
4         MULTIPLY AFW-PREMIE BY 2          instructie 3
          MOVE AFW-PREMIE TO AFW-PRINT          "      4
      ELSE
5         MULTIPLY AFW-PREMIE BY 1,5            "      5
          MOVE AFW-PREMIE TO AFW-PRINT.         "      6
      ADD AFW-PREMIE TO TOT-PREMIE.             "      7
      MOVE TOT-PREMIE TO TOT-PRINT.             "      8
      PERFORM KOP-PARAGRAAF.                    "      9
6     WRITE PRINT-REGEL.                        "     10
      GO TO EXIT-PARAGRAAF.                     "     11
```

Figuur 5 illustreert waar Combi instructies zal tussenvoegen. De pijlen in figuur 5 stellen hierbij deze instructies voor.
Bij pijl 1 wordt geteld hoeveel keer de paragraaf van het programma wordt uitgevoerd. Tevens is hiermee bepaald, het aantal keer dat conditie 1 wordt getest.
Bij pijl 2 wordt geteld hoeveel keer instructie 1 wordt uitgevoerd. Dit geeft tevens een beeld over de gebruikte (test)invoer: namelijk hoeveel posten bevatten een AFWEZIGHEID kleiner dan 10.
Bij pijl 3 wordt geteld hoeveel keer instructie 2 wordt uitgevoerd. Tevens wordt hiermee bepaald hoeveel posten een AFWEZIGHEID niet kleiner dan 10 bevatten. De tellingen verricht bij pijl 2 en pijl 3 zijn samen gelijk aan de telling bij pijl 1.
Bij pijl 4 wordt geteld hoeveel keer instructies 3 en 4 uitgevoerd zijn. Dit verschaft wederom een beeld over de inhoud van het gebruikte bestand; hierbij met betrekking tot het veld DIENST-JRN.
Bij pijl 5 wordt geteld hoeveel keer instructies 5 en 6 worden uitgevoerd. De tellingen bij de pijlen 4 en 5 vormen samen de hoeveelheid keren dat conditie 2 is getest.
Bij pijl 6 wordt geteld hoeveel keer instructies 10 en 11 worden uitgevoerd en instructie 7, 8 en 9 zijn uitgevoerd.
De tellingen bij de som van pijl 4 en 5, en pijl 6, kunnen in principe verschillen indien na de sprong vanuit instructie 9 naar KOP-PARAGRAAF niet teruggekomen wordt bij instructie 10 na uitvoering van KOP-PARAGRAAF.

Runverslag-lijst

Er zijn situaties, waarin Combi een instructie zou moeten toevoegen, maar dit niet kan zonder de COBOL-syntax of de werking van het toepassingsprogramma aan te tasten. In zo'n situatie zal Combi tijdens het modificeren de instructies, waarover Combi geen controle kan uitoefenen, op een runverslag-lijst afdrukken, waardoor de gebruiker in staat wordt gesteld deze instructies zelf door middel van source code-review te controleren.

Figuur 6



Figuur 6 illustreert drie situaties, waarbij Combi geen (volledige) controle kan uitoefenen.
Bij pijl 1 zou door Combi een instructie toegevoegd dienen te worden, ten einde vast te kunnen stellen hoeveel keer de paragraaf PREMIE-PARAGRAAF en instructie 1 worden uitgevoerd. Voor de volledigheid moet worden vermeld dat het toevoegen van een instructie syntactisch onjuist zou zijn, indien de paragraaf genoemd wordt in een ALTER-instructie.
De instructie, die toegevoegd wordt bij pijl 3, telt wel hoeveel keer aan conditie 1 wordt voldaan en hoeveel keer instructie 2 wordt uitgevoerd maar kan door de "OR"-relatie in conditie 1 niet vaststellen hoeveel posten met WISSEL gelijk aan 4 zijn verwerkt.
De instructie, die toegevoegd wordt bij pijl 4, telt hoeveel keer instructie 3 wordt uitgevoerd. Echter hierbij dient vermeld te worden, dat de verwerking door de subroutine TOTBER-ROUT niet gecontroleerd wordt, terwijl deze wel deel uitmaakt van de totale verwerking van het toepassingsprogramma.
Gedurende de modificatie door Combi van het toepassingsprogramma zullen de instructies 1 en 3 en de conditie 1 op de runverslag-lijst worden afgedrukt, voorzien van het in de regel opgenomen volgnummer van de instructie/conditie.

Uitvoer

De Combi-subroutine vervaardigt de lijst met niet uitgevoerde instructies of het aantal uitvoeringen per instructie, of zoals de figuren 3 en 4 illustreren, groepinstructies. Indien de niet uitgevoerde instructies worden afgedrukt, worden toch tellingen per instructie(groep) gemaakt. De afgedrukte instructies zijn dan de instructies, die nul maal zijn uitgevoerd.

Elke instructie, die Combi toevoegt, wordt voorzien van een volgnummer (sequence number). De door de Combi-subroutine afgedrukte nummers identificeren hiermee de toegevoegde Combi-instructies.
In de programmalijst van het gemodificeerde toepassingsprogramma, vervaardigd door het COBOL-vertaalprogramma (bij het compileren), kunnen de instructies door middel van het afgedrukte volgnummer worden teruggevonden.
De omvang van de door de Combi-subroutine vervaardigde lijst is veelal groot, doordat het gebruik van Combi het doelmatigst is bij grote, complexe programma's. Immers met source code-review kan over kleine, minder gecompliceerde programma's vaak al een voldoende beeld verkregen worden.
Alle instructies, die op de door de Combi-subroutine vervaardigde lijst staan, dienen teruggezocht te worden in de programma-listing ten behoeve van een controle op:

- volledigheid testset,
- foutieve instructies,
- overtollige instructies.

Figuur 7

```
DATUM-PARAGRAAF.
1 ->    READ DATUM-KRT AT END                  instructie 1
2 ->         DISPLAY 'GEEN DATUM-KRT'               "     2
3 ----->     STOP RUN.                              "     3
4 ->    IF   DATUM NOT NUMERIC                 conditie 1
5 ->         DISPLAY 'FOUTE DATUM-KRT'         instructie 4
6 ----->     STOP RUN.                              "     5
7 ->    MOVE DATUM TO DATUM-KOP.                    "     6
        GO TO EXIT-PARAGRAAF.                       "     7
```

Bij een normale verwerking, waarbij de datumkaart bij het initialiseren van het toepassingsprogramma wordt gelezen, gecontroleerd en akkoord bevonden, zullen de instructies 1, 6 en 7 eenmaal worden uitgevoerd en de conditie 1 eenmaal worden getest. De instructies 2, 3, 4 en 5 zullen in die situatie niet worden uitgevoerd. Combi zal dit aangeven.
Derhalve zullen bij een onderzoek naar (niet) uitgevoerde instructies de meldingen van Combi veelal snel verklaard kunnen worden, waardoor de meldingen van speciale situaties meer aandacht kunnen krijgen.

N.B.: De pijlen 3 en 6 geven aan, waar Combi de spronginstructie voor de Combi-subroutine zal toevoegen. Immers, indien instructies 2, 3, 4 en 5 zullen worden uitgevoerd, betekent dit de beëindiging van de verwerking van het toepassingsprogramma.

Figuur 8

```
**********************************************************************************
**********************      COMBI  SOURCE-LIST      ********************** PAGE  14
**********************************************************************************
```

BRANCH	PROGRAM COBOL-STATEMENT		PASSAGE
124	028150	PREMIE-PARAGRAAF.	XASST80 3025
124	028151	IF AFWEZIGHEID < 10	XASTT80 3025
125	028152	ADD 100 TO AFW-PREMIE.	XASTT80 978
126	028153	IF AFWEZIGHEID > 9	XASTT80 3025
127	028154	ADD 75 TO AFW-PREMIE.	XASTT80 2047
128	028155	IF PERS-NR = 93274	XASTT80 3025
129	028156	MULTIPLY AFW-PREMIE BY 2.	XASTT80 *******
130	028157	IF DNST-JRN > 5	XASTT80 3025
131	028158	ADD 100 TO DNST-PREMIE.	XASTT80 1903
133	028159	TOT-PARAGRAAF.	XASTT80 3025
133	028160	ADD DNST-PREMIE, AFW-PREMIE TO TOT-PREMIE.	XASTT80 3025

De sterren bij instructie 028156 geven aan, dat deze instructie nul maal is uitgevoerd.

Bij programma's, waarbij instructies vele malen worden uitgevoerd, is het interessant uit oogpunt van effeciëntie na te gaan of er delen van het programma kunnen worden herschreven, ten einde de verwerking sneller te laten plaatsvinden.

Gebruik

Het Combi-pakket kan gebruikt worden door:

- de systeemanalist/programmeur, ten einde hun programma zo snel en volledig mogelijk te kunnen testen;
- de project-manager, ten einde een beeld te kunnen vormen over de kwaliteit van de programmatuur;
- de accountant, ten einde beter inzicht te krijgen in de werking van een programma ten behoeve van een gedetailleerde beoordeling.

Slot

De meest recente versie van het Combi-pakket bezit de mogelijkheid, tijdens de uitvoering van het gemodificeerde toepassingsprogramma, een programmalijst van het ongemodificeerde toepassingsprogramma te vervaardigen, die identiek is aan de door de COBOL-compiler gegenereerde programma-afdruk.
Deze programmalijst bevat behalve het programma een vermelding per

- sectienaam,
- paragraafnaam,
- instructie,
- conditie,

het aantal malen dat zij zijn doorlopen (zie figuur 8).

Het terugzoeken van de door Combi gegeven tellingen per instructie/conditie in de gemodificeerde programmalijst, gegenereerd door de COBOL-compiler, kan hierbij achterwege blijven.
De redundante code, dit wil zeggen instructies en condities die nooit kunnen worden uitgevoerd, wordt tevens door deze faciliteit van Combi volledig ontdekt.
Dit gebeurt maar ten dele, wanneer geen gebruik van deze mogelijkheid wordt gemaakt (zie figuur 2).

De recente versies van de IBM COBOL-compiler kunnen uitgevoerd worden met een zogenaamde COUNT-optie.
Deze optie verschaft een soortgelijk resultaat als het Combi-pakket.

Verdere informatie betreffende dit pakket kunt U bij de A.C.-groep van Klynveld Kraayenhof & co verkrijgen.

Automatisering Beveiliging Controle NIEUWS

door H.C. Kocks

Automatisering

"On October 3, 1978 IBM issued a press release announcing its version of a general-purpose minicomputer, the 8100. This computer can be used as a standalone machine or as part of a larger network of machines in a distributed data processing (ddp) environment.
The significance of the announcement cannot be underestimated; it is possible computer historians will record October 3, 1978 as a turning point in the general direction of worldwide computer development. With the 8100 announcement, IBM legitimized the concept of ddp, and gave notice to the rest of the minicomputer industry that it is serious about entering this marketplace."

Aldus de introductie van een artikel van Larry Woods getiteld "IBM's 8100: first impressions", in Datamation van maart 1979. Gedeelten van deze "first impressions" willen wij U niet onthouden.

The 8100 announcement is not particularly important from the hardware viewpoint, as we will see later. What is significant is:

1. The 8100 is an IBM ddp (distributed data processing) product.
2. The 8100 is designed for online processing.
3. The 8100 comes with a reasonable set of software.
4. Complete 8100 systems will not be delivered until 1980.

HARDWARE

The hardware of the 8100 series is designed for online processing. Another interesting point about 8100 I/O implementation is that memory-to-I/O device (channel I/O) data transfer appears to use the cpu for its address translation. Channel I/O can then be expected to utilize most, if not all, of the cpu time during I/O operations. This is not what is normally defined as direct memory access (DMA), and should be considered to be a weakness in the 8100 system.

The cpu must be accessed during I/O operations because channel I/O is addressing logical memory addresses, as compared to actual memory addresses. This is necessitated because of a "unique" relocation technique used on the 8100 called dynamic address relocation.

Dynamic address relocation is an attempt to map logical, contiguous address spaces (logical memory) into physical memory (real memory) in order to maximize the utilization of real memory. This is accomplished through the use of a dynamic address translation mechanism.

IBM 8100 processor speed is not impressive and was not announced in a consistent manner.
Hopefully, there will be future 81XX processors with exciting speeds. The lack of processor speed could be compensated for with a good, efficient instruction set, which is not documented for the public at the present time.

The initial peripheral offerings for the 8100 vary widely in performance. The 8775 display terminal is an impressive display device, but it is another display device. Now IBM offers its customers the 327X and 8775 display products from DPD (Data Processing Division) and the 4978/4979 Series/1 displays from GSD (General Systems Division). A customer cannot be expected to use all of these devices, and no single piece of IBM equipment supports them all. Again another reason to consider an impending split between DPD and GSD, regardless of IBM statements to the contrary.

The fastest printer that can be attached to the 8100 is the IBM 3289 line printer model 3, which runs at 300 lpm with a 64-character set. This speed is inadequate for many RJE (remote job entry) configurations, although it may be well matched with the 9600 bps maximum transmission rate on a 8110-to-host data link.

Another interesting piece of equipment is the 2502 card reader which must connect to the 8110 through the IBM 3289 line printer! This tends to make RJE configurations rather clumsy, to say the least.

Local I/O peripheral interfacing is limited to the loop technology. Initial offerings on the 8100 fail to include the capability for connecting digital or analog sensor I/O to the processor. There are capabilities for connecting digital I/O through the digital input/digital output (DI/DO) adapter on the 3641/3642 reporting terminal. This restriction does not allow the flexibility that is offered by other minicomputer manufacturers. It will still be necessary to use other vendors' equipment to interface into the sensor environment unless IBM offers the capability for interfacing this type of device into the 8100.

SOFTWARE

Software is the strength of the 8100 systems. Two operating systems (DPPX and DPCX), various resident software subsystems, and optional host support packages, are what is going to make the IBM 8100 a viable product.

The Distributed Processing Control Executive (DPCX) is an operating system which is offered to the existing IBM 3790 customer as a migration system. DPCX will allow the 3790 user to phase out of his 3790 and into the 8100 environment.

The Distributed Processing Programming Executive (DPPX) is the general-usage operating system for the IBM 8100. DPPX offers a higher degree of flexibility in system design than DPCX, and will require more user education in order to take advantage of that flexibility.

Two standard high-level language offerings (FORTRAN and COBOL) can be compiled on the 8100 under the DPPX operating system. Both compilers generate reentrant code. This feature allows for the creation of efficient online applications. Reentrant code allows multiple users to execute a single copy of application code. This makes much better use of memory and reduces or eliminates the necessity for paging code in and out of the processor (a feature which is not available on the DPPX operating system).

DPPX data access methods include an indexed access methode (IAM) data set. This facility is utilized by DTMS, the Data Base and Transaction Management System, in order to provide what IBM calls "Data Base Management". This feature of DTMS provides for check-pointing of IAM transactions and subsequent backout, if necessary. Although the term "data base management" is used, it should not be confused with a data base management system (DBMS), such as IMS.

The absence of a true DBMX implementation on the IBM 8100 is considered to be a weakness by many potential users. Furthermore, existing IMS users prefer that a "mini-IMS" be made available on the 8100. Although there is obvious value in supporting a DBMS on the 8100, an eight-index indexed access method can provide the capability for designing reasonably complex data bases. IAM should also be relatively less complicated in its implementation which, in turn, will provide a more reliable 8100 installation. There is much to be said for simplicity.

The most disappointing aspect of the IBM 8100 announcement is the delivery schedule. Although the first deliveries of hardware are due this August, much of the software is not scheduled for availability until early 1980.

PROBLEM DETERMINATION IS THE USER'S RESPONSIBILITY

Software installation and subsequent problem determination will be the responsibility of the customer.

In order to support the user in his debugging efforts, the DPPX operating system will provide a sizable number of diagnostic tools, including memory dumps, system traces and error logs (both software and hardware).

Hardware installation can also be done by the user. Again, an early IBM press release notes: "For example, portions of the new system can be installed by users - with a set of easy-to-follow directions - in much the same way as a basic household stero system might be set up".

This feature is defined as Customer Set-Up (CSU) by IBM. It is available on the 8130/8140 processors and the 8101 storage and I/O unit. CSU is explained by IBM sales people as meaning that the customer can uncreate his machine, roll it into place, release the disk unit (with a handy lever in the front of the unit), plug it in, and turn it on.

User installation is new even to the minicomputer industry.
Most minicomputer manufacturers require their systems be installed by customer engineers; the CE's also execute the diagnostic tests. This installation procedure is followed so that the equipment is reliable enough to be able to allow the customer to "bring it up" on his own.

The marketing plan for the 8100 is to sell boxes, both hardware and software. The customer assembles his configuration(s) from the shopping list of offerings.

PLANNING FOR THE NON-MIS USER

IBM has not ignored the end-user in its 8100 announcements. The 8100 under the DPPX operating system is a standalone computer. It has the capabilities necessary to allow an 8100 user to develop complete data base oriented systems without the use of a central computer.

For users who do not understand procedural computer languages, IBM has introduced an 8100-resident programming aid, the Development Management System (DMS). DMS is described as: "A sophisticated programming aid that uses English-language statements to lead operators through the creation of new system application programs. The system automatically translates the operator's instruction into COBOL, allowing persons without data processing backgrounds to create programs in COBOL".

Datamation, maart 1979

De noodzaak van een goede automatiseringsorganisatie en van maatregelen met betrekking tot de handhaving van de continuïteit van de gegevensverwerking wordt meestal onderkend. Dat een minderheid (i.c. computerspecialisten) in onze samenleving ondanks de getroffen maatregelen aan de meerderheid haar wil kan opleggen blijkt uit het volgende artikel. De uitdrukking "kennis is macht" heeft kennelijk nog niet aan betekenis verloren.

Samenleving de dupe van staking onder computerspecialisten

Dat we in ons dagelijks leven eigenlijk niet meer buiten de computer kunnen hebben we de afgelopen periode kunnen zien bij onze Engelse buren. Daar werden zo ongeveer alle overheids- en semi-overheidsinstellingen lamgelegd door een staking onder de computerspecialisten. De staking, die werd georganiseerd door de twee ambtenarenbonden - de Society of Civil and Public Servants en de Civil and Public Services Association - had tot doel een loonsverhoging te bewerkstelligen van 30% en was in de eerste instantie gericht tegen de computerinstallaties van de overheid.

Zo kwam het computersysteem van de douane in Southend tot stilstand, hetgeen resulteerde in het feit dat niet langer de BTW kon worden berekend op de ingevoerde goederen. Alleen dit al zou de Engelse regering, wanneer de staking zou voortduren, een aanzienlijke som geld kosten. Daarnaast worden

vooral ook de uitbetalingen van subsidies aan boeren en industrie in gevaar gebracht. Dat een dergelijke staking grote gevolgen zou kunnen hebben werd reeds vorig jaar gemeld in een rapport van de Engelse ambtenarenbond (Civil Service Dept) over administratieve automatisering bij de centrale overheid. In dat rapport werd reeds gewezen op de uitwerking van een actie van com puterspecialisten met betrekking tot de samenleving. De staking die nu werd georganiseerd omvatte ruim 1300 medewerkers en het is beangstigend te weten dat een relatief klein aantal in staat is een deel van de economie plat te leggen.
Het is duidelijk dat door de enorme snelle ontwikkeling van de technologie samenlevingen als die in Engeland en bij ons geheel afhankelijk worden van een gering aantal mensen, die de kennis, vaardigheid en de sleutelpositie beheren binnen de voor de economie zo vitale instellingen. Sommige publikaties spreken van een "ontzagwekkende macht over het wel en wee van hun landgenoten", welke dan weer leidt tot een "zware verantwoordelijkheid" ten opzichte van die landgenoten.
Een logische gevolgtrekking daarvan zou zijn, dat van zo'n machtspositie geen misbruik mag worden gemaakt. Dat zou in de praktijk betekenen dat stakingen van overheidsfunctionarissen op vitale technologische posten niet geduld behoeven te worden. Mocht er te zijner tijd een toekenning plaatsvinden van het stakingsrecht aan ambtenaren, dan zal eerst moeten worden overwogen wat de eventuele gevolgen daarvan zouden kunnen zijn, ten einde een chaos te voorkomen.

De Automatisering-gids, 8 maart 1979

Beveiliging

"Gegevensbeveiliging niet honderd procent", is de kop van een onlangs in de Automatisering-gids verschenen artikel. Om de gegevensbeveiliging echter te verbeteren, wordt nog veel onderzoek verricht. Onderstaand artikel gaat in op de verfijning/verbetering van een in 1976 ontworpen "handtekeningverificatiesysteem".

Algoritme ontdekt valsheid in geschrifte

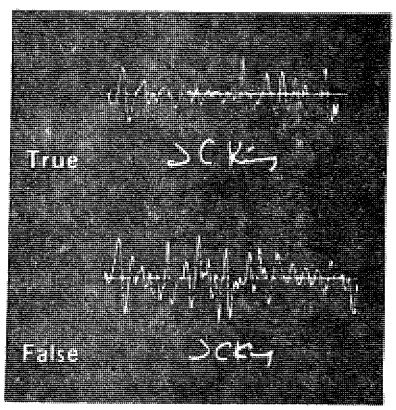
Na vier jaar van intensief speurwerk hebben onderzoekers van IBM's Research Centrum te Yorktown Heights een systeem ontworpen, dat bijna perfect een valse van een echte handtekening kan onderscheiden. De machine doet dat door de bewegingskarakteristieken van pendruk en -versnellingen met de orignelen in een data base met elkaar te vergelijken. Het experimentele handtekeningverificatiesysteem kan een aanvulling worden op bestaande beveiligingsmethoden tegen ongeoorloofde toegang tot gegevensbestanden.

Handtekeningen die door het nieuwe systeem op hun echtheid worden getoetst, moeten geschreven worden met een speciale pen, waarin kleine versnellingsmeters en een drukdetector zijn aangebracht. Tijdens het schrijven van een handtekening wordt informatie betreffende de druk op en de

versnellingen langs het papier verwerkt door een kleine computer, die gebruik maakt van een speciaal voor dit doel ontworpen algoritme. Druk en versnellingen tijdens het plaatsen van een handtekening zijn voor ieder individu even uniek als zijn vingerafdrukken. Daardoor herkent het systeem in verreweg de meeste gevallen een valse handtekening, die uiterlijk niet in karakter verschilt van de echte. Zelfs wordt een mislukte handtekening vaak nog als echt herkend, wanneer die wordt geplaatst door de authentieke persoon.

Onlangs is het verificatiesysteem in de praktijk getest. Hierbij waren 248 vrijwilligers betrokken. Van de 2.958 echte handtekeningen werden er door het systeem 2.909 (98,3%) ook als echt herkend.
De twee niet herkende valse handtekeningen worden door Dr. Noel M. Herbst, één van de ontwerpers van het systeem, toevalstreffers genoemd.
Dat een iets groter percentage echte handtekeningen als "vals" werd herkend, is volgens Dr. Herbst niet zo belangrijk. In mogelijke toekomstige toepassingen, waarin zeer hoge eisen aan de beveiliging worden gesteld, kan een volgende poging om de identiteit met een handtekening te bewijzen, als regel worden toegestaan. Veel belangrijker is dat het systeem na twee of drie mislukte pogingen kan besluiten de toegang tot een computersysteem of een data base te blokkeren, omdat er kennelijk met valse handtekeningen wordt gewerkt.

In een eerdere versie van dit handtekeningverificatiesysteem, waarop Dr. Herbst en John H. Morrissey in 1976 patent verwierven, werd alleen de versnelling van de pen gemeten en verwerkt. Daarna is het systeem verbeterd om het percentage juist herkende falsificaties verder op te voeren.
Behalve de introductie van de druksensor in de pen, betrof deze verbetering de uitwerking van een nieuw beslissingsalgoritme. Deze algoritme meet en vergelijkt zeer gedetailleerd de overeenkomsten in druk- en versnellingskarakteristieken en gaat voorbij aan grote verschillen, die vaak optreden in varianten op echte handtekeningen.



Voorbeeld van een handtekening, die door de computer als echt wordt herkend (boven) en een zorgvuldig geïmiteerde falsificatie. Op basis van het patroon herkent de computer de handtekening (onder) als onecht.

In het Financieele Dagblad van 26 april 1979 is het volgende stuk opgenomen.

"Dief benut kennis van computersysteem

De Amerikaanse computerdeskundige Stanley Rifkin is veroordeeld tot acht jaar gevangenisstraf voor het stelen van $ 10.200.000 van een Amerikaanse bank. De rechter wees het aanbod van de verdachte af, voor de politie lezingen te mogen houden over computeroplichting.
De 32-jarige Rifkin stal het geld - het was één van de grootste diefstallen uit de Amerikaanse geschiedenis - van de Security Pacific National Bank door het plaatsen van een telefoontje van 10 dollarcent naar de bank, oktober vorig jaar. Hij maakte gebruik van zijn kennis over het computersysteem van de bank om geld over te maken naar Zwitserland, kennis die hij had opgedaan als mede-ontwerper van het systeem."

De diefstal kwam aan het licht doordat Security Pacific alle telefonische transacties op tape vastlegt. De stem van Rifkin werd herkend. Een aantal zaken, die niet uit het stukje in het FD blijken, doch wel in een artikel in EDPACS van januari 1979 zijn opgenomen, willen wij U niet onthouden.

Rifkin kreeg vrij eenvoudig toegang tot de computerzaal omdat hij daar een bekend figuur was wegens daar verricht "consulting work".
Om de telefonische overboeking te plegen, verzamelde hij daar drie vitale gegevens, namelijk:

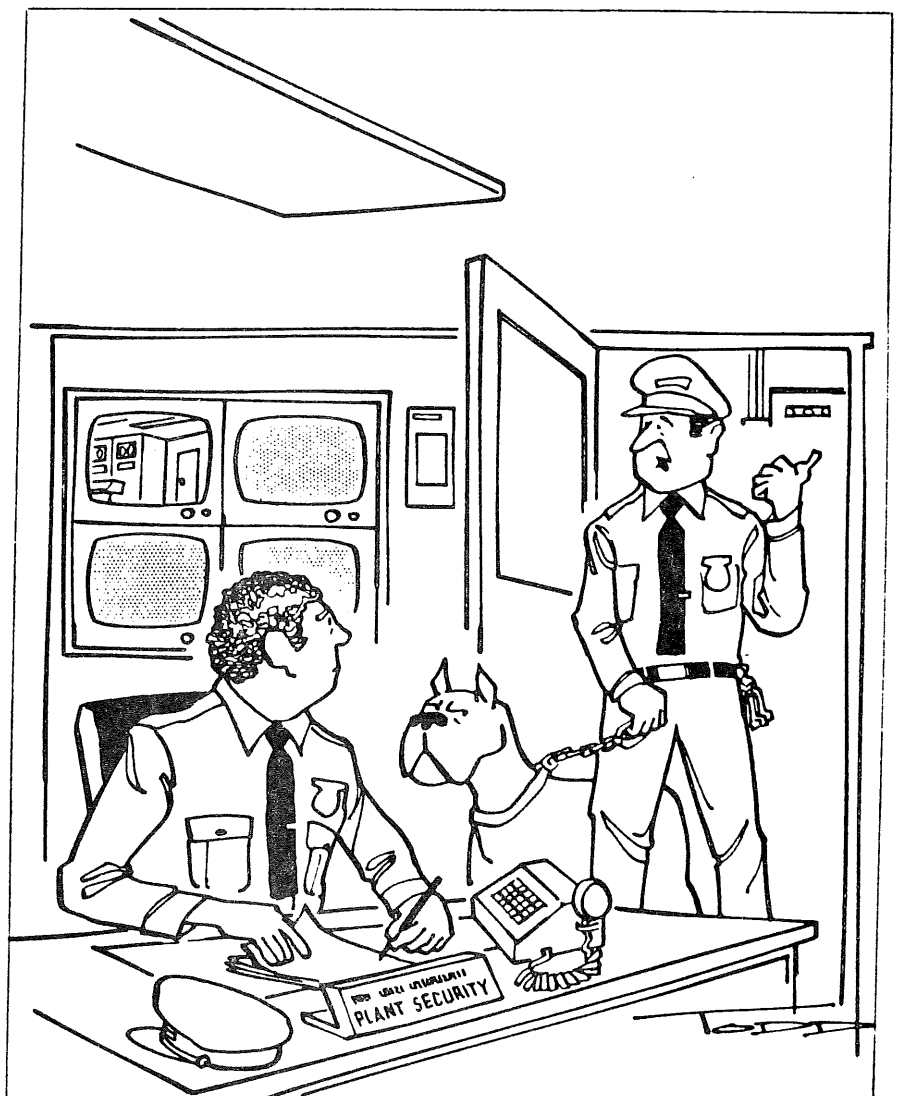
- The security code number used to authenticate each day's transfer orders, changing each day.
- The personal code used by one of the bank's officers to identify himself to the system.
- The number of an account that had a large balance on deposit.

Aan het einde van de dag, als employees moe zijn, belde hij de afdeling "telefonische orders" onder gebruik van de hem bekende gegevens en bracht een overboeking van $ 10.000.000 naar een bank in Zürich tot stand.

Opmerkelijk uit oogpunt van security, audit en control, zijn achteraf de volgende punten.

1. It took eight days before anyone noticed that a crime had been committed. A more professional or better prepared criminal could have disappeared off the face of the earth in that time frame. Sensitive financial systems should be able to detect large irregularities in a reasonably short time. For example, in this case, perhaps bank officers should have been required to give a next-day approval signature on all telephone transfer orders that used their identification code.
2. A three-level access control scheme should be quite effective. Many installations do very well with two levels of control. However, in the absence of effective physical access controls, no system access control plan will provide an acceptable level of security. Mr Rifkin

had no reason to be in the wire transfer area. He should never have been allowed to enter it. If he did gain entry, he should have been challenged and asked to leave. Employees who work in sensitive areas should receive periodic reminders of the importance of controls and the need for their active participation in the enforcement of security provisions.
There is no way to design a system that will be totally resistant to assault by a determined, intelligent, and skilled individual like Mr Rifkin. However, the systems that are designed and implemented should take full advantage of available and cost-effective security and control procedures.



"Someone's stolen three of our closed-circuit cameras!"

© DATAMATION

Controle

Nieuw is de problematiek inzake de controle van "Program Maintenance" niet, wel belangrijk! Program Maintenance verdient - in toenemende mate - de aandacht van zowel de interne als de externe accountant, omdat Program Maintenance:

- een steeds groter deel van de activiteiten van een automatiseringsafdeling gaat uitmaken;
- uit oogpunt van controle één van de meest kwetsbare activiteiten van een automatiseringsafdeling is; de goedkeurings- en testprocedures zijn vaak minder strak geregeld;
- meestal door minder ervaren programmeurs wordt verricht, waardoor de kans op fouten wordt vergroot.

In het volgende artikel - met een naschrift van H. Weiss - gaat M.I. Sobol in op de controle van Program Maintenance.

Control of Program Maintenance

From a control standpoint, one of the most vulnerable activities in any data processing department is program maintenance. Production programs, which appear to be both error-free and in accordance with user requirements, are often modified to meet new requirements. During this maintenance process, strict administrative and organizational controls must be employed to insure that modifications are properly requested, approved, coded, tested, documented, and authorized for production.
These controls will also help to prevent unauthorized, and potentially fraudulent, changes from being made.

Each step in the maintenance process, from initiation through implementation, requires its own procedures or guidelines to protect the integrity of the computer application. This article will review the program maintenance process, identify areas of potential exposure, and describe the controls which can be used to minimize the associated risks. Prudent data processing executives will want to set up adequate controls within their installation. Internal and external auditors will also be interested in these controls when they are evaluating the strengths and weaknesses of a particular installation.

Request for change

A program "work order", "maintenance request", or "work request" is initiated by a user. The multipart request must receive proper user department authorization. The form should contain the following information:

- Date of request
- Required completion date
- Reason for the program change
- Impact of the change, if known, on other parts of system
- Detailed description of the change, including, if applicable, sample reports illustrating the impact of the modification
- Signature of requesting user
- Authorized signatures indicating user approval.

This information will help the data processing personnel understand the nature of the request, estimate manpower and time requirements, and insure that the request has been thoroughly considered and properly approved by the user.
One copy of the request is retained by the user, the remaining copies are forwarded to data processing where they are logged and assigned a control number. Installations that charge data processing costs to user departments will analyze the request and estimate the cost to complete the modification. This will include the cost of programming, testing, and documentation. A predetermined cut-off point, say $ 1,000, will be used to determine major requests which will be sent back to the initiating user for another approval before additional work will be performed. These administrative controls minimize the likelihood of unauthorized or economically unsound requests being implemented.

Programming

Once the work request is accepted, it becomes a project within the maintenance department. It will be assigned to a programmer for analysis and coding. These programmers, in order to do their job, must have access to system and program documentation as well as program source code. As a result, they have most of the tools necessary to analyze a series of programs, design and fraud, and insert unauthorized code without arousing suspicion. To reduce this risk, some additional controls, discussed below, are required.

Testing

Comprehensive unit and volume testing of modified programs is required before the revision can be accepted. These tests are equal in importance to the testing procedures applied during the initial development of a system. Program testing cannot be overemphasized. A maintenance change in one module of a system may often have far-reaching effects on many other modules. If testing is limited to the changed code without an attempt to validate the remaining system components, it may lead to a major system failure.

Documentation

Before approving and implementing the modification, a "Program Revision Record" form should be made part of the program's or module's documentation. This revision record should contain full details of the change and should be supported by an updated source listing and a copy of the original change request. Some source librarian software packages provide an audit trail of program changes. If this is available, it should be included in the documentation of the change. Program maintenance documentation is often overlooked because of the pressure to get the work out. This approach will serve only to increase overall program maintenance time requirements because inaccurate and incomplete documentation will hinder future maintenance activity.

Controls

The program maintenance process previously described illustrates some potential areas of vulnerability; for example, undocumented changes and untested programs. Controls which can be used to insure that maintenance changes are properly applied and that will reduce exposure to error or fraud are:

- User authorization

  Written evidence that the requested changes has been properly initiated and approved by the user department reduces the possibility of unauthorized requests being submitted through normal maintenance channels. Each request should require two user signatures, the initiator's and a supervisor's.

- Organization control

  Supervisory personnel within the maintenance group must review and approve all changes in writing. Although a supervisor may not have time to review every line of modified code, a formal approval mechanism will help deter attempts at improper manipulation.

- Independent application of program changes

  The integrity of source code changes can be improved if programmers must submit their debugged source program changes to a quality assurance function. The quality assurance group reviews the changes and applies them to copies of the production source programs. They also maintain an audit trail of changes and include it in the documentation. While the QA group cannot review and understand all program changes, their presence minimizes the likelihood that unauthorized code will be inserted in production copies of source programs. (Note: If a data processing organization does not have a formal QA group, the functions assigned to it in this article should be assumed by some other entity independent of the maintenance programming activity.)

- Independent acceptance testing

  Acceptance testing of the modified system should be performed by the QA group. To implement this assurance technique, the QA staff must develop a base case test file which thoroughly exercises a system's logic. After maintenance changes have been applied, the updated system is tested using this base case test file. Any discrepancies between the results from prior base case tests and those from the changed system must be attributed to maintenance activity. This approach can require extensive time and manpower commitments. It is advisable, therefore, to limit base case testing of low priority systems. For example, such a system might be retested quarterly, while a high priority, high risk system would be tested after each maintenance change. Used judiciously, base case testing will create a very high degree of confidence in the integrity of maintenance changes.

- Documentation control

  Quality Assurance should be responsible for reviewing documentation updates caused by program maintenance. Prior to implementation of the modified system, all required documentation changes, including a revised program narrative and a program revision record, must be submitted and approved by QA.
  Without this formal acceptance of updated documentation, the system modification should not considered to be complete.

- Library control

  QA personnel should be the only people authorized to apply program changes to and catalog newly updated programs on production libraries. This will deter programmers from modifying production source programs and prevent them from compiling and replacing a production load module with another program. These restrictions on source and load libraries are essential if any of the other controls described in this article are going to be effective.
  Internal auditors are usually responsible for insuring that the QA department is functioning properly and effectively. Because of its responsibility for controlling program maintenance, the QA group's activity must be subject to in-depth review on an annual basis.

Conclusion

The program maintenance procedures and controls suggested herein will contribute to the security and integrity of any data processing installation. Management must accept the ultimate responsibility for insuring that these procedures are implemented and kept in proper working order.

Postscript (by Harold Weiss)

Even if organizations do not charge back data processing costs to user department, it is still a good discipline to cost out changes. It furnishes the basis for a potentially stronger control procedure than that described by Mike Sobol.

It would seem desirable to have more signatures on "major" requests. Having the user as the sole authorization for changes, as some organizations do, can lead to catastrophic results; for example, a serious deterioration in response time of the system. Should not the systems manager and/or higher data processing management approve the change if sufficient cost or systems impact were involved? Should not higher levels of management sign off on major modifications or require modifications of this magnitude to be treated as new systems?

The internal audit function is also a candidate for an approval signature. There are four alternatives to the audit interface with program maintenance:

1. At present most internal auditors are not involved in any way. This is not good practice. As more and more business procedures are implemented by computer programs, including control procedures, the integrity of these programs must become a serious audit concern.

2. The greatest audit involvement would be to sign off in advance of implementation of changes. Some audit groups have done this for many years, but it is not feasible for most because of staff limitations and higher priorities. The auditors may not have sufficient familiarity with the systems being changed. If they were not involved with what went into the systems during the original development, why should they worry about changes in these systems?
   A large organization with volatile systems may have 200 or more changes a week. As changes accumulated into a backlog, there would be pressure for meaningless audit signoffs so as to avoid delays in implementation. Hence, if this magnitude of audit interface is not really practical, it should be avoided in order not to create management illusions about control. However, as computerized business systems become more critical, complex, and hazardous, some progressive managements are requiring prior audit signature for both new systems implementation and their modification. A 1976 survey of medium and large organizations with relatively heavy commitment to EDP auditing showed 19 percent required auditors to sign off on program modifications (EDPACS, September 1977, page 13).

3. The audit function may merely be marked for copies of the modification approval documents. If these will be utilized, even for review on a sampled basis, then the cost is justified. If papers merely pile up in the audit area, there will be no deterrent to unauthorized changes. The data processing staff will soon be aware if there is no real audit use of these documents. Auditors with an occational need to review approval documents can have access to data processing and user records.

4. Most practical for most internal audit groups today, and the required minimum of involvement, is the traditional audit approach. This is to establish that adequate control procedures, such as those described by Mike Sobol, are in place and to make periodic compliance tests to assure that the controls are working. This takes much less audit effort than the second alternative. Testing is necessary because there have been numerous abuses of potentially good systems of program maintenance control. Compliance techniques could include interviews, code comparisons, review of librarian software reports, analysis of logs, flowcharting software, code review, etc. Procedure 4 can be used along with numbers 2 or 3.

EDPACS, februari 1979

AC 205 Data processing cost reduction & control - D. Brandon
Van Nostrand 1977 (Engels, 190 blz.)

Trefwoord: B 59 (Kosten en baten van systemen)

Data processing wordt in de laatste jaren omvattender, waardoor de kosten van onderhoud stijgen. Dit boek verschaft een aantal technieken aan het management om deze kosten te beheersen en mogelijk te verlagen. Naast een analyse van de problemen geeft het een groot aantal suggesties voor kostenvermindering (met checklists).

AC 208 Risk analysis and control, a guide for the DP manager - K.K. Wong
NCC Publ. 1977 (Engels, 144 blz.)

Trefwoord: B 48 (Risicobepaling)

"As companies become increasingly dependent upon the use of computers, so the security of those computer systems takes on much greater importance. In many organisations the risks to which the computer is exposed could cause the failure of the organisation if the risks are not adequately guarded against.
Dealing with these and other security problems is now fast becoming a new branch of management - risk management - requiring new skills. This book gives guidance on methods of risk identification, the probability and impact of risks, approaches to risk evaluation, and suggests a policy for survival."

Dit boek maakt deel uit van de serie "Computer Security" van het National Computing Centre, als verslag van het werk dat op dit gebied de laatste jaren is verricht. Het omvat de hoofdstukken Risk Identification, Risk Evaluation en Risk Control.