INTERAKTIE MET PERSOONLIJKE INFORMATIESYSTEMEN

P. van Oostrum

RUU-CS-83-3

januari 1983

Rijksuniversiteit Utrecht

Vakgroep informatica

Princetonplein 5
Postbus 80.002
3508 TA Utrecht
Telefoon 030–53 1454
The Netherlands

INTERAKTIE MET PERSOONLIJKE INFORMATIESYSTEMEN

P. van Oostrum

Technical Report RUU-CS-83-3

januari 1983

Department of Computer Science

University of Utrecht

P.O. Box 80.002

3508 TA Utrecht

the Netherlands

INTERAKTIE MET PERSOONLIJKE INFORMATIESYSTEMEN

P. van Oostrum

Vakgroep Informatica
Rijksuniversiteit te Utrecht
Postbus 80.002, 3508 TA Utrecht.

Abstract.

In dit rapport wordt een overzicht gegeven van de belangrijkste problemen en vraagstellingen m.b.t. het persoonlijk, interaktief gebruik van informatiesystemen. Ten eerste worden verschillende aspekten van informatiebeheer (tekstverwerking, databases, elektronische post, e.d.) besproken en de verschillende interaktiemogelijkheden genoemd. Uit deze bespreking blijkt dat er veel overeenkomsten zijn tussen de verschillende genoemde gebieden en technieken. Verschillende principes die op één gebied (bijv. tekstverwerking) gangbaar zijn blijken op een ander gebied (bijv. databases) waardevol te zijn. Daarna wordt een voorstel geformuleerd om tot een integratie van deze tot nu toe veelal afzonderlijk beschikbare informatiekategorieeën en interaktiemogelijkheden te komen. Tenslotte wordt een inventarisatie gegeven van de research die nodig is om tot een dergelijke integratie te komen.

Keywords: tekstverwerking, databases, informatiesysteem, interaktie, editors, kantoorautomatisering, ergonomie.

0.0 Inleiding.

Door de snelle prijsdaling en capaciteitsvergroting van computerapparatuur zal in de nabije toekomst een groot gedeelte van het niet computer-georienteerde publiek te maken krijgen met het gebruik van computers. In het bijzonder de opkomst van de microcomputer maakt het mogelijk in een organisatie een groot deel van het personeel te voorzien van personal computers (werkstations).

Tot nu toe zijn het voornamelijk programmeurs en andere technisch georienteerde personen, die met de computer weten "om te gaan", terwijl meer administratief gerichte personen zich beperken tot het gebruiken van kant en klare toepassingen, die door meer deskundigen bedacht en opgezet zijn. De laatste gebruikers fungeren hierbij vaker als intelligent I/O bestanddeel van het systeem (data entry), dan als iemand die de computer gebruikt als

instrument om zijn *) problemen op te lossen. Bij de toename van het aantal kleine computersystemen zal het onmogelijk zijn om alle te bedenken toepassingen door programmeurs te laten ontwikkelen. De gebruiker zelf zal hier mogelijkheden moeten krijgen om op een creatieve manier met het systeem om te gaan, en een groot scala van toepassingen op eenvoudige en natuurlijke manier te formuleren. Idealiter zal het gebruik van een persoonlijk werkstation moeten kunnen worden vergeleken met het gebruik van apparatuur als telefoon, typemachine, copieerapparaat, wasmachine e.d. Deze wensen geven aanleiding tot onderzoek naar nieuwe interaktiemogelijkheden met apparatuur en softwaresystemen. In dit rapport beperken we ons tot het gebruik van de computer als informatiesysteem.

In hoofdstuk 1 geven we een beknopt overzicht van de problematiek van het maken van en werken met een persoonlijk informatiesysteem (d.w.z. een informatiesysteem dat voor een individu van belang is).

In hoofdstuk 2 worden de belangrijkste aspekten van het interaktief werken met computers besproken. De belangrijkste kenmerken van interaktief werk worden in (2.1) genoemd, en vergeleken met niet-interaktief werk. Hierbij zal blijken dat veel zg. interaktief werk in feite nog veel kenmerken van niet-interaktieve verwerking in zich heeft. In (2.2) geven we een overzicht van de belangrijkste kategorieeen werk die voor interaktieve verwerking in aanmerking komen en bovendien op het terrein van informatiesystemen liggen of hieraan grenzen. Tenslotte geeft (2.3) een overzicht van de technieken die beschikbaar zijn om interaktie met een systeem mogelijk te maken, of te bevorderen.

In hoofdstuk 3 wordt geprobeerd om de principes die in hoofdstuk 2 naar voren zijn gekomen, te integreren. Verschillende technieken die in gebruik zijn bij een bepaalde toepassing, blijken ook bij andere toepassingen nuttig gebruikt te kunnen worden. Hierbij krijgen we een zicht op een mogelijk informatiesysteem, waarin diverse aspekten, zoals tekstverwerking, toegang tot databases, etc. op een uniforme manier aanwezig zijn.

Hoofdstuk 4 geeft een aanzet om de struktuur van een dergelijk geintegreerd informatiesysteem te formuleren. Zowel de benodigde informatiestrukturen als de operatoren die hiervoor nodig zijn, worden hier genoemd. Aan de hand hiervan wordt een geintegreerde ´information editor´ voorgesteld. Deze wordt vergeleken met projekten die op een aantal universiteiten lopen.

---

*) In dit stuk wordt uitsluitend in de mannelijke vorm aan de computergebruiker gerefereerd. Hiermee is geen onderwaardering van de vrouwelijke gebruikers bedoeld. De lezer wordt verzocht waar dit van toepassing is, ´hij/zij´ resp. ´zijn/haar´ te lezen in plaats van ´hij´ resp. ´zijn´.

Hoofdstuk 5 geeft tenslotte een scenario van de researchaktiviteiten die voor een dergelijk systeem nodig zijn, of er een raakvlak mee hebben.

In dit rapport wordt voornamelijk ingegaan op de software aspekten van interaktieve systemen. Specifieke hardware aspekten (zoals de layout van toetsenborden, vereisten voor beeldschermen, e.d. worden hier niet of slechts summier aangestipt. Hiermee is niet gezegd dat deze aspekten niet belangrijk zijn, maar zij vallen buiten het kader van dit stuk.

1.0 Probleemdefinitie.

Onder een informatiesysteem verstaan we een geheel van hardware en software (niet noodzakelijk op basis van een computer) voor het opbergen, organiseren en opzoeken van informatie. Informatie is meer dan een verzameling gegevens: een belangrijk aspekt van informatie wordt gevormd door de relaties tussen de diverse gegevens, m.a.w. door de organisatie ervan. Een ongeorganiseerde verzameling informatie heeft de neiging om meer verwarring dan inzicht te schenken; we kunnen dan beter spreken van "disinformatie". Het is daarom belangrijk om goede struktureringsmogelijkheden in een informatiesysteem te hebben. Deze struktuur zal dan weer dienst kunnen doen bij het terugvinden van de informatie. Er kunnen diverse redenen zijn om informatie op te slaan in een geautomatiseerd informatiesysteem in plaats van in een "papieren" informatiesysteem (kaartsysteem, mappen e.d.).

We zullen in de rest van dit rapport alleen over geautomatiseerde informatiesystemen spreken, en dan het adjectief geautomatiseerd weglaten. De belangrijkste voordelen van zo'n informatiesysteem zijn:

- Het gemakkelijker beheren van grote hoeveelheden informatie
- Associatieve toegang (toegang op basis van de inhoud)
- Toegang via de struktuur
- Hergebruik en selektie van bestaande informatie

We kunnen ruwweg de volgende kategorieen informatie onderscheiden:

- sterk gestruktureerde (databases)
- ongestruktureerde (tekst)

. numerieke (rekenmodellen)

. grafische (plaatjes)

In dit rapport zullen we ons niet bezig houden met grafische informatie, hoewel sommige geformuleerde principes zich op natuurlijke wijze hierop laten overdragen. De overige kategorieën worden in het algemeen in de toepassingen vrij streng gescheiden (resp. database systemen, tekstverwerking, en numerieke programmatuur), hoewel er de laatste tijd een tendens is om deze gebieden bij elkaar te brengen. We zullen verderop gemeenschappelijke kenmerken van deze kategorieeen zoeken en zo proberen een geintegreerd concept van informatieverwerking te formuleren. Hierbij zullen we ook proberen concepten die in een kategorie vruchtbaar blijken te zijn, toe te passen in de andere kategorieen.

Bij grote informatiesystemen moet het systeem bedacht en opgezet worden door systeemontwerpers en programmeurs. De gebruiker kan zijn wensen kenbaar maken tijdens het ontwerp, maar bij het gebruik van het systeem is hij onderworpen aan datgene wat in het systeem is aangebracht. Dit beperkt vaak het creatieve gebruik van het systeem als men zich buiten de oorspronkelijke opzet begeeft. Dit in tegenstelling tot het werken met niet-automatische systemen waar men de vrijheid heeft om eigen strukturen te bedenken wanneer zich onverwachte problemen voordoen.

Het maken van een klein (ad-hoc) informatiesysteem voor een plotseling opkomende toepassing is vaak geen eenvoudige zaak. Meestal is hiervoor de tussenkomst van een programmeur nodig. Met de komst van moderne query-talen als SQL, Datatrieve, Query-by-Example e.d. is deze taak aanzienlijk vereenvoudigd, hoewel deze talen nog lang niet in alle aspekten als gebruikersvriendelijk ervaren worden [Green78].

Met de komst van de personal computer, die goedkoop en in grote aantallen beschikbaar komt, wordt het probleem des te dringender. De prijs van de hardware is hier zo laag geworden dat het niet meer doenlijk is om een programmeur te hebben, die voor elke toepassing een programma maakt. Bovendien wil men snel een toepassing kunnen proberen wil men het systeem werkelijk als een persoonlijk apparaat gebruiken.

Het verst gevorderd m.b.t. gebruikersvriendelijkheid in deze kategorie zijn de tekstverwerkers. Het werken hiermee sluit nauw aan bij het werken met de traditionele typemachine. De basishandelingen die men op een tekstverwerker kan uitvoeren zijn dezelfde als die van een normale typemachine, terwijl het apparaat daarnaast de mogelijkheid biedt om geavanceerde operaties uit te voeren, zoals het verplaatsen van tekst (komt overeen met traditionele knip en plak operaties), en het opzoeken van tekst (nieuwe faciliteit). Bij de betere tekstverwerkers heeft men getracht om zo nauw mogelijk aan te sluiten bij het arsenaal van begrippen en operaties die

de gebruiker (e.g. secretaresse) kent.

2.0 Interaktie.

In dit hoofdstuk bespreken we de kenmerken van interaktief werken met informatiesystemen. Hiertoe worden in eerste instantie de verschillen tussen interaktieve en niet-interaktieve systemen geanalyseerd en voorwaarden voor effektieve interaktie geformuleerd (2.1). Daarna volgt een overzicht van de soorten toepassingen in het vlak van interaktieve informatiesystemen (2.2). Tenslotte inventariseren we de technieken, mechanismen en apparaten, die de gebruiker ten dienste staan om met een interaktief systeem om te gaan.

2.1 Voorwaarden voor interaktie.

In het laatste decennium is veel onderzoek gedaan naar de psychologische aspekten van het gebruik van computers (o.a. gebruikersvriendelijkheid). Veel van deze experimenten hebben het probleem dat zij zich beperken tot een zeer klein deelgebied van het te onderzoeken terrein, zowel wat betreft het toepassingsgebied, als de groep van gebruikers (proefpersonen). Hierdoor (en door gebruik van verschillende onderzoeksmethoden) komt men nogal eens tot tegengestelde konklusies. Toch zijn er langzamerhand wel enige algemeen aanvaarde principes gedestilleerd. Een goed overzicht van de resultaten van deze onderzoekingen vindt men in [Shnei80].

2.1.1 Batchverwerking. - Tot voor kort was het meeste computergebruik op grote systemen "batch" georienteerd. Hierbij heeft de gebruiker nauwelijks of geen interaktie met de computer of met zijn programma's. Vooral in een administratieve omgeving kan dit betekenen dat de uiteindelijke gebruiker zelfs niets van de uitvoering merkt: hij levert zijn invoerdata op formulieren in, deze worden voor hem geponst (resp. op diskette gezet), iemand anders voert het in de computer in, en hij krijgt de uiteindelijke uitvoer in de vorm van lijsten terug. Eventuele fouten in de invoer moeten hierna hersteld en de gekorrigeerde invoer opnieuw aan de computer aangeboden worden. Belangrijke nadelen van deze werkwijze zijn:

1. De grote "turn-around" tijd, d.w.z. de tijd die verstrijkt tussen het aanbieden van een stuk werk (job) en het terugkrijgen van de resultaten ervan. Vooral m.b.t. het korrikeren van fouten is een

korte turn-around-tijd gewenst.

2. De voortdurende kans op inkonsistentie van de data in het computersysteem. Een van de oorzaken hiervoor is de bovengenoemde turn-around-tijd, en het effekt hiervan op foutieve invoer.

3. De moeilijkheid om creatief met de computer om te gaan.

Het laatste punt is belangrijk genoeg om bij stil te staan. Wanneer er een hoge mate van interaktie mogelijk is tussen een gebruiker en de computer (resp. software), dan heeft dit tot gevolg dat:

a) de gebruiker kan experimenteren met het systeem, om te zien wat de gevolgen zijn van verschillende aannamen.

b) het gedachtenproces van de gebruiker gestimuleerd en gericht kan worden door de uitkomst van zijn opdrachten.

c) de gebruiker voortdurend kan beschikken over de gegevens die hij nodig heeft bij het oplossen van zijn probleem.

Het is duidelijk dat in veel traditionele, vast liggende toepassingen deze voordelen minder belangrijk zijn, terwijl voor veel nieuw opkomende toepassingen (bijv. beleidsondersteunende) deze van groot belang zijn. De mogelijkheden van interaktieve persoonlijke systemen hebben in dit opzicht zelfs al tot nieuwe toepassingen geleid (bijv. tekstverwerkers, elektronisch kladblok, zie 2.2)

2.1.2 Interaktieve verwerking. - Bij volledig interaktieve verwerking heeft de gebruiker maximale interaktie mogelijkheden met het systeem (= de combinatie van hardware en software). We hebben hierboven al enkele malen het voorbeeld van de tekstverwerker genoemd. Het kenmerk van een interaktief systeem is dat er atomaire handelingen zijn, die door de gebruiker beinvloed kunnen worden, en waarvan de gebruiker ook onmiddelijk het resultaat ziet. Welke handelingen als atomair beschouwd moeten worden hangt hierbij af van de toepassing, de ervaring en ook de psychische struktuur van de gebruiker.

Uit de cognitieve psychologie zijn o.a de volgende vereisten bekend:

1. De vrijheid om kreatief bezig te zijn.

   De gebruiker moet hierbij niet in een keurslijf gewrongen worden, maar hij moet een model van zijn probleem op kunnen bouwen dat (ook visueel) aansluit bij zijn eigen voorstelling. Hij moet kunnen voortbouwen op de methodes die hij zonder het systeem

gebruikt zou hebben (bijv. op papier), maar het systeem moet ook uitnodigen om op meer creatieve wijze met de eigen mogelijkheden ervan om te gaan. Bij onderzoekingen is gebleken dat gebruikers in het algemeen de neiging hebben om maar beperkt van nieuwe mogelijkheden gebruik te maken, in het bijzonder wanneer het systeem niet voldoende uitnodigend is. De kreatieve mogelijkheden worden vooral beknot wanneer de mogelijkheden die het systeem biedt, moeilijk te achterhalen of te herinneren zijn, en wanneer de (visuele) terugkoppeling beperkt is.

Bennet merkt op dat interaktieve voorzieningen kunnen leiden tot veranderingen in de probleemoplossende gewoontes van de gebruiker, en dat de manier van presentatie van de informatie voor de gebruiker kan leiden tot nieuwe manieren om over de logische struktuur te denken [Benn72]. Een goed interaktief systeem zal daarom moeten uitnodigen tot het gebruiken van nieuwe informatiestrukturen, m.a.w. het moet de gebruiker niet te veel vastleggen op een eenmaal gekozen informatiestruktuur.

Een belangrijke voorwaarde om kreatief met een systeem te kunnen omgaan is verder het niet-destruktief zijn van de handelingen van de gebruiker. Dat wil zeggen dat wat de gebruiker ook doet, hij altijd weer terug kan naar zijn oorspronkelijke uitgangspunt. Wanneer dit niet het geval is, zullen er situaties zijn, waarin de gebruiker zich geremd voelt om de mogelijkheden van het systeem ten volle te gebruiken, uit angst dat er onomkeerbare akties uitgevoerd zullen worden.

2. Ondersteuning van het geheugen van de gebruiker.

Zoals in de vorige alinea al aangeduid is, is een van de moeilijkheden van interaktieve systemen vaak het herinneren van de mogelijkheden die het systeem biedt, en het bedenken van wat men gedaan heeft, en wat de resultaten van de akties zijn. Om het geheugen van de gebruiker te ondersteunen zijn diverse technieken in gebruik (Help funkties, menu's, etc. zie 2.3). Ook de terugkoppeling van visuele informatie is hierbij vaak nuttig. In het algemeen blijkt dat twee dimensionale, grafische informatie voor veel personen betere resultaten oplevert dan sequentiele, een-dimensionale [Russ79], hoewel hierbij ook sprake is van verschil t.g.v. verschillende orientatie van personen (o.a. dominantie van linker resp. rechter hersenhelft, zie o.a. [Shnei80]).

Dit leidt tot de volgende belangrijke kenmerken voor een interaktief systeem:

1. Grote mate van kontrole door de gebruiker (gebruiker bepaalt, machine gehoorzaamt).

2. Atomaire operaties (zo mogelijk door de gebruiker te definieren).

3. Direkte terugkoppeling van het resultaat van operaties ("What You See Is What You Get").

4. Ondersteuning van het geheugen van de gebruiker.

5. De mogelijkheid om het effekt van operaties weer ongedaan te maken.

Pas wanneer aan deze eisen is voldaan kan men spreken over een instrument dat effektief is voor het oplossen van problemen.

2.1.3 Semi-interaktieve systemen. - Wanneer we interaktieve systemen analyseren op bovengenoemde kriteria, dan moeten we konkluderen dat veel zg. interaktieve systemen nog veel batch georienteerde eigenschappen hebben. Als voorbeeld nemen we de methode van tekstverwerking die op veel timesharing systemen gebruikelijk is. Hierbij worden de volgende fasen onderscheiden:

1. - Maak een zg. "markup" file aan met een editor. Deze file bevat de tekst, die in het uiteindelijke dokument moet komen, met ertussen zg. formatteringskommando's, die aangeven hoe de uiteindelijke tekst eruit zal komen te zien. De editor is meestal min of meer interaktief.

2. - Verwerk deze "markup" file m.b.v. een tekstformatter.

3. - Bekijk het resultaat (op een terminal of op een afdruk) en zoek de fouten op. Verbeter de fouten op de "markup" file door weer naar punt 1 te gaan.

Ook bij het interaktief ontwikkelen van programma's zien we vaak een soortgelijke cyclus: edit - compilatie - link - run ... waardoor het foutvrij maken van een programma vaak een moeizame klus wordt. Systemen waarbij deze operaties geintegreerd zijn, hebben dan ook terecht een grote populariteit.

We kunnen konstateren dat in deze gevallen in feite sprake is van een batchgeorienteerde verwerking in een interaktieve omgeving. Deze methode heeft ook veel van de voornoemde problemen van batchverwerking. We kunnen deze verwerking dan ook beter semi-interaktief noemen. Als belangrijkste oorzaken van het niet echt interaktief zijn, kunnen we noemen:

- Te grote atomaire operaties.

- Te grote responsetijd van de operaties.

2.2 Interaktieve werkzaamheden.

De belangrijkste kategorieën interaktief werk zijn:

programmeren

* tekstverwerking

* database operaties

* electronic mail en telekonferentie

* planning en modellering

graphics

computer assisted instruktion (CAI)

computer aided design (CAD)

Hierbij is met (*) aangegeven die onderwerpen die voor dit rapport van belang zijn, daar zij met informatiesystemen te maken hebben.

Er zijn diverse raakvlakken tussen de verschillende gebieden, de belangrijkste zullen verderop besproken worden. We noemen hier nog even terzijde het belang van graphics voor praktisch alle andere toepassingen (voorstelling van visuele modellen, maar ook het inbedden van grafische informatie in tekstverwerkingsdokumenten en het opslaan van grafische informatie in databases). Verder moet hier ook nog genoemd worden het belang van de "recreatieve informatica". Het maken en bestuderen van computerspelletjes kan een belangrijk hulpmiddel zijn voor het onderzoek van interaktieve systemen, omdat hier in het algemeen een extreem grote gebruikersvriendelijkheid vereist is. Het is te verwachten dat de toekomstige generatie van computergebruikers hiermee ervaring zal hebben opgedaan en van professionele computersystemen hetzelfde gemak zal verwachten.

2.2.1 Tekstverwerking. - Tekstverwerking is een van de belangrijkste aktiviteiten van computergebruikers. Sommige onderzoekers spreken van 50-70%. Met de komst van de speciale tekstverwerker (tekstverwerker) op het kantoor is het aantal personen dat hiermee gekonfronteerd wordt, sterk toegenomen.

Traditioneel is tekstverwerking gesplitst in twee aktiviteiten: teksteditiing en tekstformattering.

Teksteditiing is het invoeren en korrigeren van eenvoudige tekst. Deze wordt hierbij i.h.a. beschouwd als een lineaire string van symbolen, soms met als extra struktuur een regelindeling. De belangrijkste operaties die men hierbij kan uitvoeren zijn: weglaten, toevoegen, opzoeken, wijzigen en verplaatsen van tekst.

Tekstformattering is het "opmaken" van de tekst, d.w.z. indelen in pagina's, hoofdstukken, alinea's, het maken van voetnoten, kortom de complete layout verzorging.

Op timesharing systemen heeft men i.h.a. interaktieve teksteditors. Deze worden behalve voor het verzorgen van invoer voor formatters, ook gebruikt voor het maken van programma's, invoerdata, e.d. waarbij de uiterlijke verschijningsvorm van de tekst direkt in de editor verzorgd wordt. Deze editors bedienen zich nogal eens van een kryptische notatie voor de verschillende operaties, terwijl het resultaat van de operaties zichtbaar gemaakt moet worden d.m.v. een of ander print kommando. Deze editors zijn daarom vaak moeilijk te gebruiken voor "leken".

Modernere teksteditors (full-screen editors) laten een gedeelte van de file op een beeldscherm zien, zodat wijzigingen direkt zichtbaar worden. sommige gebruiken funktietoetsen (zie 2.3.2) voor veel gebruikte operaties. Hoewel het idee van de fullscreen editor al oud is (Krachtige fullscreen editors worden al beschreven in [VDam71, Iron72]), is er pas recent een grote belangstelling voor deze vorm van editing. Dit dient waarschijnlijk toegeschreven te worden aan de relatief late beschikbaarheid van de beeldschermterminals en het grotere computervermogen dat hiervoor nodig is.

Teksteditiing mag zich verheugen in grote interesse onder onderzoekers. Veel publicaties en onderzoeken houden zich echter nog steeds bezig met het maken van nieuwe editors gebaseerd op de oude (semi-interaktieve, kryptische) principes. Meestal zijn hierbij dan andere aspekten, waar het onderzoek op gericht is (bijv. portabiliteit). Meer recente publicaties over full-screen editors vindt men o.a. in [Snee78], waar een 4-kleuren display editor besproken wordt, met konklusies over interaktief gebruik, [McLeod77] (screen editor op Unix, waar superioriteit t.o.v. konventionele editors gekonkludeerd wordt), [Charl81] (line editor en screen editor geintegreerd), [Carg81].

Er zijn kwantitatieve resultaten verkregen met betrekking tot de tijd die een bepaalde edit-klus kost, en het aantal fouten dat gebruikers maken in een bepaalde klus.

Card, Moran en Newell [Card80] hebben geprobeerd de tijd te voorspellen is, die nodig is voor het uitvoeren van bepaalde operaties. Volgens hun onderzoek is deze tijd een lineaire funktie van het aantal elementaire operaties die nodig is om de aktie uit te voeren. Als elementaire operaties nemen zij: toetsaanslagen, aanwijsoperaties op het scherm, handbewegingen tussen verschillende delen van de terminal (bijv. toetsenbord, scherm, joystick), het trekken van lijnen (bij grafische invoer), mentale voorbereidingen (denktijd), en de responsetijd van het systeem. Dit model is alleen geverifieerd onder beperkte omstandigheden: de gebruiker moet getraind (expert) zijn, de taak moet een routinematige, kleine eenheidstaak zijn, de methode moet van te voren bekend zijn, en er moeten geen fouten gemaakt worden. Zij vergeleken drie tekst editors (twee lijn editors en een scherm editor), drie grafische editors en een aantal operating systeem interfaces. Het model voorspelde de tijden met een nauwkeurigheid van ca. 20%. De gebruikte schermeditor gaf aanzienlijk betere resultaten dan de lijn editors, voornamelijk doordat het aantal toetsaanslagen voor dezelfde operatie veel lager was. Hoewel de reikwijdte van dit onderzoek beperkt is (tijd is niet het enig belangrijke, de kans op het maken van fouten draagt bijv. aanzienlijk bij tot het al dan niet gebruikersvriendelijk zijn van een systeem), zijn deze resultaten interessant om te overwegen bij het ontwerp van een informatiesysteem. Wat betreft de gebruikersinteraktie kunnen we hieruit konkluderen dat het van belang is om o.a. het aantal benodigde toetsaanslagen en handbewegingen te beperken.

Een recent overzicht van de onderzoeksresultaten op het gebied van gebruikersvriendelijkheid van editors vindt men in [Embl81]. Een nieuw overzicht van de huidige stand van zaken op het gebied van editors vindt men in een speciaal nummer van Computing Surveys [Meyr82]. Hier worden lijn-georienteerde, karakter-georienteerde en scherm-georienteerde editors besproken, de laatste zowel met als zonder formatteringsmogelijkheden. Dit nummer bevat tevens een overzichtsartikel over tekstformattering [Furut82].

Het wetenschappelijk onderzoek naar geintegreerde editors/formatters is nog niet zo lang op gang [Coul76, Hamm81, Cham81, Strom81, Stall81, Allen81]. Hoewel kommercieel al vele tekstverwerkers te koop zijn, wordt hierover weinig gepubliceerd (waarschijnlijk uit konkurrentieoverwegingen). Een goed overzicht van de belangrijkste rapporten op dit gebied vindt men in [Reid81], terwijl in [SIGPL81] recente resultaten te vinden zijn.

2.2.2 Database operaties. - Het opslaan van gegevens in een database is natuurlijk een van de belangrijkste bezigheden voor een informatiesysteem. Hierbij zal het vaak gaan om sterk gestruktureerde data, hoewel ook ongestruktureerde tekst een aandeel kan hebben, bijv. bij het beheren van literatuurverzamelingen.

Bij het werken met databases is er een duidelijk verschil tussen het definieren van een database en de toegang tot bestaande databases. Voor het laatste is er tegenwoordig een breed scala van querytalen beschikbaar (bijv. SQL, Datatrieve, QBE). Meestal zijn deze gebaseerd op het relationele model, hoewel ook pogingen zijn ondernomen om soortgelijke mogelijkheden voor andere datamodellen te creeren (bijv. als relationele laag op een DBTG model). Het definieren van een database is i.h.a. een moeizame bezigheid. In de relationele query talen zijn meestal voorzieningen aanwezig om een database op te zetten, maar voor o.a. het hierarchische en het netwerk model is men aangewezen op moeilijk te gebruiken DDL compilers e.d. Het opzetten van een goede database is, algemeen gesproken, een gekwalificeerde taak voor een database beheerder, maar in een persoonlijk informatiesysteem moet er een mogelijkheid zijn om eenvoudige bestanden te definieren. Deze operatie dient dan conceptueel even simpel te zijn als de gewone toegang tot de database en als (bijv.) het maken van een goede layout voor een tekstdokument. Hierbij dient de gebruiker de keus te hebben uit een redelijke verzameling datamodellen, liefst met de mogelijkheid tot integratie van concepten uit de relationele, netwerk en hierarchische strukturen. Zo zal voor het voorstellen van 1:1 of m:n relaties het relationele model meestal de voorkeur hebben, terwijl voor 1:n relaties een (DBTG) set-struktuur beter kan zijn. Het is belangrijk hierbij zo nauw mogelijk aan te sluiten bij de voorstelling van de gebruiker.
De meest gebruikte operaties op databases zijn:

. Toevoegen, weglaten en opzoeken van data.

. Selektie (subset) van een database

. Projektie (weglaten van attributen)

. Join (combineren van data via gemeenschappelijke attributen)

Reisner [Reis77] beschrijft hoe de querytaal SQL zich ontwikkeld heeft n.a.v. psychologische experimenten met de talen SEQUEL (voorloper van SQL) en SQUARE. Uit deze experimenten komen o.a. de volgende aanbevelingen voor een query taal:

. De taal moet gelaagd zijn, d.w.z. er moet een serie van eenvoudige subtalen mogelijk zijn.

. De syntax moet aansluiten bij de natuurlijke taal.

. Konsistentie in de naamgeving van data-elementen.

. Konsistentie in het gebruik van aanhalingstekens.

. Zo mogelijk spellingskorrektie en acceptatie van synoniemen.

Bij het laatste punt bleek een algemene moeilijkheid het gebruik van enkelvoud / meervoud voor relatie- en attribuutnamen. De natuurlijke formulering gebruikt deze vaak door elkaar in verschillende kontexten. Zo kan men bijv. een bestand (relatie) "werknemers" hebben en de query willen uitvoeren: vind de "werknemer" (enkelvoud), die etc... Dit soort "fouten" kan opgevangen worden als de query processor deze woorden als synoniemen accepteert. Overigens is het gebruik van dit soort query talen, ondanks de oppervlakkige gelijkenis met natuurlijke talen, toch nogal wiskundig georienteerd, vooral wanneer men ingewikkelde queries heeft (nesting).

Query-by-Example (QBE) is een systeem dat probeert bij de gebruiker aan te sluiten door een twee-dimensionale voorstelling van relaties op een beeldscherm te geven (zie bijv. [Zloof77a, Date81]). Uit experimenten is gebleken dat QBE gemakkelijk te leren is maar dat het enig begrip veronderstelt van de eerste orde predikatenlogika [Zloof78]. Vergelijkingen tussen QBE, SQL en een op de relationele algebra gebaseerde taal geven aan dat QBE superieur is m.b.t. query korrektheid, vertrouwen van de gebruikers (in de korrektheid van hun queries) en snelheid, waarbij de subjecten studenten waren [Green78]. Het belangrijkste probleem waar men op stuitte bij het gebruik was het aanbrengen van "links" tussen verschillende relaties (i.e. de relationele join operator).

QBE is ook uitgebouwd tot een systeem (SBA - System for Business Automation) waarbij de gebruiker op soortgelijke wijze een totale administratieve toepassing kan definieren door het computersysteem "voorbeelden" te geven van de te volgen procedures [Zloof77b]. Een verdere uitbouw naar meer algemene kantoorprocedures, zoals electronic mail heeft plaatsgevonden in OBE (Office by Example, [Zloof81]).

Ook zijn er ontwikkelingen op het gebied van meer grafisch georienteerde query-talen zoals Cupid en Foral, en van het grafisch weergeven van database informatie [Herot80].

Reisner [Reis81] vergelijkt een aantal onderzoeken op het gebied van gebruikersvriendelijkheid van query talen. Zij konkludeert dat de experimentele resultaten op dit gebied vaak moeilijk interpreteerbaar zijn. Men moet voorzichtig zijn met het trekken van konklusies. Ook zijn resultaten van verschillende experimenten vaak onvergelijkbaar. De punten waar beter op gelet zou moeten worden zijn: de details van het experiment

(wat heeft men eigenlijk onderzocht), de statistische signifikantie van de resultaten, en de reikwijdte van de konklusies (deze worden te gauw gegeneraliseerd).

2.2.3 Electronic mail en telekonferentie. - Dit is een gebied dat nauw verwant is met tekstverwerking, daar electronic mail meestal zal plaats vinden met dokumenten die door tekstverwerking aangemaakt zijn. Voor het administreren van de post zal men een soort database systeem nodig hebben. Telekonferentiesystemen hebben met tekstverwerking gemeen dat zij met ongestruktureerde tekst werken, terwijl daarnaast de behoefte kan bestaan om tijdens de konferentie inzage in (nog te distribueren) dokumenten te hebben. Behalve de normale edit-mogelijkheden is er bij elektronische post behoefte aan administratieve funkties, zoals het organiseren van de post in rubrieken, opzoeken op sleutelwoorden, terugmelding van ontvangst (resp. lezing), het doorsturen van post met annotaties e.d. Palme [Palm79] beschrijft een telekonferentiesysteem, waarbij de systeemfunkties d.m.v. menu's gekozen kunnen worden, terwijl on-line assistentie voor de gebruiker aanwezig is d.m.v. een "help" funktie. Hij beschrijft het belang van de mogelijkheid van het systeem om zich (automatisch) aan te passen aan de ervaring van de gebruiker.

2.2.4 Planning en modellering. - In dit verband verstaan we onder "planning en modellering" interaktieve software voor het doorrekenen van getallenmateriaal in eenvoudige rekenmodellen (ook wel elektronisch kladblok genoemd). Een model is hierbij een matrix (eventueel driedimensionaal), waarbij relaties tussen elementen, rijen of kolommen gedefinieerd kunnen worden. Sommige elementen worden dan als onafhankelijke grootheden beschouwd, terwijl andere gedefinieerd kunnen worden d.m.v. (arithmetische) formules. Wanneer een onafhankelijk element gewijzigd wordt, brengt het systeem automatisch de veroorzaakte wijzigingen in afhankelijke elementen aan. Doordat op ieder moment de waarde van alle elementen (of een deelverzameling) op het beeldscherm aanwezig is, leent dit systeem zich uitermate goed voor het experimenteren met numerieke modellen. De grondlegger van deze systemen was het programma "VisiCalc" (handelsmerk van VisiCorp) [Rams80]. De operaties die bij een elektronisch kladblok mogelijk zijn, zijn de normale editmogelijkheden op de data-elementen, het weglaten en invoegen van rijen en kolommen, en editing van de formules, die de relaties tussen elementen aangeven. Eventuele uitbreidingen zouden bijv. kunnen zijn het sorteren op kolommen of rijen, statistische bewerkingen e.d.

2.3 Interaktiemechanismen.

Terwijl in de batchomgeving de voornaamste in- en uitvoermedia de ponskaart en de (papieren) lijst zijn, is er bij het interaktieve werk een keur van mogelijkheden. De meest gebruikte apparaten en methodieken zijn:

- terminals (hardcopy/beeldscherm)
- grafische beeldschermen
- grafische input (tablet, digitizer)
- formulieren
- funktietoetsen
- menu's
- aanwijsmechanismen (joystick, muis)
- kommandotalen
- natuurlijke taal
- spraak
- help funkties

Een aantal hiervan zullen we hieronder meer gedetailleerd bespreken.

2.3.1 Formulieren. - Simulaties van (papieren) formulieren op een beeldscherm worden veel gebruikt in administratieve omgevingen voor scherp gedefinieerde taken. De gebruiker moet (mag) hierbij bepaalde vakjes van zijn beeldscherm invullen, terwijl de rest van het beeldscherm dient als referentie of informatie en beschermd is (d.w.z. niet gewijzigd mag worden). De gebruiker krijgt hierdoor een overzichtelijke presentatie van de informatie en van zijn parametermogelijkheden, die nauw aansluit bij wat hij gewend is, of waar hij zijn invoer vandaan haalt. Meestal kan hij de vakjes in een willekeurige volgorde invullen en bijv. achteraf nog wijzigingen in een ingevuld vakje aanbrengen voor de hele invoer wordt vrijgegeven. De vakjes in het formulier kunnen meestal van attributen worden voorzien, zoals: het moet numeriek (resp. alfabetisch) zijn, het aantal symbolen dat moet worden ingevuld, rechts of links aangeschoven, het mag leeg gelaten worden, etc. Deze methode is zeer gebruikelijk bij data entry, waar een leeg

formulier telkens moet worden ingevuld, en waarbij de inhoud (na kontrole) wordt weggeschreven in een database. Voor het opzoeken van gegevens in een file kan dit nuttig zijn als er niet teveel zoekkenmerken zijn. De gebruiker specificeert dan welk zoekkenmerk (sleutel) gebruikt moet worden door het betreffende vakje in te vullen. Ook het maken van wijzigingen in een database kan gemakkelijk zijn m.b.v. formulieren, als de wijzigingen klein zijn. Het systeem kan dan een formulier op het scherm laten zien met de originele gegevens, terwijl de gebruiker dan gebruik maakt van edit mogelijkheden om kleine wijzigingen in het formulier aan te brengen.

Formulieren worden meestal vanuit applicatieprogramma's bestuurd, of soms d.m.v. een query taal *). De operaties die men op een formulier kan uitvoeren, zijn eenvoudige edit funkties, en het springen naar velden. Voor het maken van een formulier heeft men een speciale editor nodig die ook attributen van velden kan definieren. De velden die men op een formulier kan definieren zijn meestal van vaste lengte, terwijl ook vaak een formulier maximaal een beeldscherm groot kan zijn. Sommige terminals hebben bijzondere voorzieningen voor het werken met formulieren (bijv. beschermde velden, ingebouwde kontroles).

Formulieren als gegeneraliseerde interface tussen een gebruiker en een computersysteem worden ook voorgesteld door het Codasyl End User Facilities Committee [Lefk79].

Aan de Universiteit van Toronto ontwikkelt men een systeem voor kantoorautomatisering, gebaseerd op het gebruik van formulieren als algemene interface van de gebruiker met het systeem [Tsich82]. Hierbij worden de formulieren zowel gebruikt voor het weergeven van bij elkaar horende informatie (records uit een database), als voor het specificeren van operaties die op de informatie moet worden uitgevoerd. Dit systeem is in veel opzichten vergelijkbaar met SBA [Zloof77b] en OBE [Zloof81]. Een interessant aspekt van het systeem is dat het behalve textuele informatie o.a. ook spraak kan opslaan en (in beperkte mate) manipuleren.

2.3.2 Funktietoetsen. - Dit zijn speciale toetsen op een terminal waarmee veel gebruikte operaties snel kunnen worden aangegeven. Bij tekstverwerkers treft men i.h.a. toetsen aan voor het weglaten van een letter, woord of regel, voor het onderstrepen van tekst, positioneren van de cursor, etc. Soms worden ook gewone toetsen gebruikt in combinatie met een speciale "funktie" toets. Het is ook mogelijk speciale funktietoetsen naast of onder het beeldscherm te hebben, waarbij de betekenis van de toetsen op het scherm gezet wordt (in feite een combinatie van menu en funktietoetsen).

---

*) Formulieren als direkte interface naar een database zijn m.i. nog nergens gebruikt. QBE kan als rudimentaire vorm hiervan beschouwd worden.

Funktietoetsen kunnen eigenlijk ook als een hardware implementatie van menu's opgevat worden. Een nadeel van funktietoetsen is dat ze bij grote aantallen onoverzichtelijk worden, en dat het moeilijk is er een variabele betekenis aan te geven i.v.m. de naamgeving.

Uit het reeds genoemde onderzoek van Card, Moran en Newell [Card80] zou men kunnen afleiden dat funktietoetsen voordelig zijn omdat ze het aantal benodigde toetsaanslagen voor veel gebruikte operaties verminderen.

2.3.3 Menu's. - Een menu is een lijst waaruit de gebruiker een keuze kan maken. Menu's worden veel gebruikt voor het kiezen van operaties of subsystemen, wanneer een niet al te grote, vaste keuze mogelijk is. Ze worden ook vaak in samenhang met formulieren gebruikt.
De manier waarop de keuze gemaakt wordt, kan zijn:

1. door een nummer in te typen, dat de keuze aangeeft.

2. door de symbolische naam van een keuze in te typen of een afkorting ervan.

3. door een keuze op het scherm aan te wijzen, bijv. met een lichtpen, aanraking van het scherm, e.d.

Het grote voordeel van menu's is dat zij het geheugen van de gebruiker ontlasten: hij hoeft niet meer alle mogelijkheden van het systeem te onthouden, maar hij kan bij de keuze de lijst van mogelijkheden nagaan en via associatie de juiste keuze maken. In dit verband zijn de keuzemogelijkheden 2 en 3 ook beter dan 1, wanneer de gebruiker niet zelf kan kiezen welke van deze methoden hij gebruikt. Belangrijk is ook dat de gebruiker niet overwelmd wordt door een te grote keuze. Wanneer een menu te groot dreigt te worden is het beter het op te splitsen in een hierarchische verzameling submenu's [Palm79]. Deze methode wordt ook bij Viewdata systemen gebruikt. Voor een geoefende gebruiker kan een menu gauw vervelend worden, vooral wanneer de snelheid waarmee het beeldscherm opgebouwd wordt, te langzaam is. Wanneer de snelheid groot genoeg is kan de gebruiker in de meeste gevallen het menu gewoon negeren, en direkt zijn keuze intypen. Anders is het aan te bevelen om de gebruiker de keuze te geven om het menu of gedeelten ervan uit te schakelen en alleen op verzoek (bijv. met een funktietoets) het weer te genereren.

Het XEROX Smalltalk systeem is een geavanceerd systeem waarbij menu's op een grafisch beeldscherm de belangrijkste interface met de software vormen [Kay77, BYTE81].

Het ZOG systeem, ontwikkeld aan de Carnegie-Mellon Universiteit [Rober79] is gebaseerd op menu's voor de totale interaktie met de gebruiker. Dit systeem heeft als belangrijke kenmerken o.a. dat het een snelle responsetijd garandeert (0.1 sec voor het vertonen van een menu), dat de menu's in een groot menu-netwerk zijn opgenomen, de menu keuze kan gebeuren via een aanraakscherm of via het toetsenbord en dat het systeem gebruikt kan worden als algemene interface, ook met gebruikersprogramma's. Zie ook [Snod80] voor een op de principes van ZOG gebaseerde interface op een microcomputer. Deze combineert de ZOG menu's met formulieren. Deze laatste worden gebruikt voor de toegang tot databases, waarbij elk veld van het scherm formulier overeenkomt met een veld uit de database.

Er zijn intussen diverse kommerciele systemen (vooral personal computers en tekstverwerkers) beschikbaar die als belangrijkste interface het menu gebruiken.

2.3.4 Aanwijsmechanismen. - Er zijn verschillende methoden in gebruik om posities op een beeldscherm aan te wijzen. Op een eenvoudige beeldschermterminal kan men met speciale funktietoetsen (pijltjes) de cursor bewegen. De tijd die men hiervoor nodig heeft hangt af van het aantal stappen dat men moet doen. Moderne tekstverwerkers hebben vaak funktietoetsen om het begin van een woord, regel, zin of alinea op te zoeken, waardoor men in grotere stappen (dus sneller) het doel kan bereiken.

Speciale instrumenten voor het aanwijzen van posities op het scherm zijn o.a. joystick (stuurknuppel), lichtpen, muis (een over de tafel rijdend muisvormig instrument, waarmee een cursor beweging wordt aangegeven, en drukgevoelig scherm (aanwijzen met de vinger). Aanwijzen op een scherm is een belangrijke operatie voor het selekteren van informatie (bijv. een tekstgedeelte bij een teksteditor) en voor het kiezen in menu's. Het Smalltalk systeem [BYTE81] gebruikt een muis met drukknopjes voor deze operaties, waardoor het menu gebruik het karakter van (uiterst flexibele) funktietoetsen krijgt.

Een aantal onderzoekers heeft experimenten met sommige van deze instrumenten uitgevoerd, waaruit een voorkeur voor de muis bleek, m.n. wat betreft de snelheid van het positioneren. Deze experimenten hebben echter alle slechts een deel van de mogelijke apparatuur onderzocht. Er zijn bijv. geen gepubliceerde resultaten, die het gebruik van een muis (als in Smalltalk) vergelijken met een aanraakscherm (als in ZOG).

2.3.5 Kommandotalen. - Waar menu's te bewerkelijk zijn of waar de keuze van de opdrachten te groot of te complex is om met menu's, funktietoetsen of formulieren te werken, daar is men aangewezen op kommandotalen. Voorbeelden van kommandotalen zijn JCL (Job Control Language, de besturingstaal voor een operating systeem), kommandotalen van teksteditors en formatters en database query talen. De oudere versies van deze talen gebruiken meestal zeer kryptische notaties, terwijl tegenwoordig meer op natuurlijke taal lijkende kommando's gebruikt worden (bijv. SQL). Vooral editors gebruiken vaak kryptische notaties, omdat dit de mogelijkheid geeft om korte kommando's te hebben, die niet teveel typewerk van de gebruiker vragen. Hierdoor worden de kommando's vaak wel onleesbaar of onbegrijpelijk (zelfs voor ervaren gebruikers). Voorbeelden van editors met zo'n kommandotaal zijn TECO en de Unix editor [Meyr82].

Ledgard [Ledg80] beschrijft een experiment waarbij een teksteditor met kryptische notatie wordt vergeleken met een die dezelfde mogelijkheden heeft, maar waarbij een pseudo-natuurlijke taal gebruikt wordt. Hieruit wordt gekonkludeerd dat de pseudo-natuurlijke taal superieur is, hoewel anderen deze konklusie aanvechten [CACM81]. Hierbij moet opgemerkt worden dat de pseudo-natuurlijke talen vaak slechts een oppervlakkige gelijkenis met echte natuurlijke talen hebben. Zij zijn dermate beperkt en strikt in syntax dat het voor onervaren gebruikers vaak moeilijk blijft ze te gebruiken.

Een belangrijk probleem wordt veroorzaakt doordat de verschillende komponenten van een systeem verschillende kommandotalen gebruiken, waardoor de gebruiker gedwongen is voortdurend gedachtensprongen te maken bij de overgang van één subsysteem naar een ander (bijv. van een tekstverwerker naar een database systeem). Voor een vruchtbaar systeem is het nodig dat konsekwent in het hele systeem dezelfde notaties en operatoren gebruikt worden voor dezelfde funkties. In dit rapport zal ook een indikatie gegeven worden van mogelijke gemeenschappelijke operatoren voor de verschillende subsystemen (zie 4.3).

2.3.6 Natuurlijke talen. - Hoewel diverse onderzoekers al jaren proberen om een natuurlijke taal interface met software te realiseren (evt. in combinatie met stem-invoer), zijn de resultaten nog niet geweldig. Codd beschrijft een systeem (RENDEZVOUS) voor interaktie met een relationeel database systeem, waarbij het systeem de gebruiker om additionele informatie vraagt, wanneer een query niet eenduidig is [Codd74, Codd78]. Hierbij maakt het dan gebruik van menu's. De grootste moeilijkheid bij het gebruik van natuurlijke taal lijkt wel de grote mate van dubbelzinnigheid die er in een natuurlijke taal aanwezig is. Het goed begrijpen van natuurlijke taal vereist een "referentiekader", d.w.z. kennis van dat gedeelte van de wereld waarover gesproken wordt. De toepassingen van natuurlijke taal zijn dan ook

voornamelijk beperkt tot specifieke toepassingsgebieden, ze lenen zich op dit moment nog niet zo goed tot het gebruik in algemeen toepasbare systemen. Het onderzoek naar het gebruik van natuurlijke talen in computersystemen ligt op het gebied van de zg. kunstmatige intelligentie (AI). Het zg. vijfde generatie computersysteem, dat men in Japan hoopt te ontwikkelen, zal voor een groot gedeelte met behulp van natuurlijke talen benaderd moeten kunnen worden. *)

Voor een goed overzicht van de problemen op dit gebied zie men [Shnei80]. Voorlopig lijkt het onwaarschijnlijk dat natuurlijke taal een belangrijk aandeel in de interaktie met computersystemen zal hebben.

2.3.7 Spraak. - Het gebruik van spraak als kommunikatiemedium met computers staat nog in de kinderschoenen. Het verstgevorderd is men op het gebied van stemuitvoer, maar de kwaliteit van algemeen bruikbare stem-generatie is niet erg goed. Wanneer het gaat om een beperkt vocabulair is heel wat meer mogelijk, zij het ten koste van veel computergeheugen. Op het gebied van steminvoer heeft men alleen nog apparatuur die ofwel een redelijke woordenschat van een persoon kan herkennen of een zeer beperkte woordenschat van meerdere personen. Er is echter wel een grote ontwikkeling op dit gebied.

2.3.8 Help funkties. - Voor een beginnende gebruiker is het belangrijk dat het computersysteem adekwate informatie geeft als er problemen zijn (bijv. als de gebruiker een fout maakt). Teveel systemen geven onbegrijpelijke foutmeldingen. Daarnaast is het belangrijk dat de gebruiker een indicatie krijgt hoe het probleem is op te lossen. Ook voor de gevorderde gebruiker is dit nodig, i.h.b. wanneer hij gebruik maakt van ongebruikelijke funkties.

Uit een onderzoek aan de Carnegie-Mellon Universiteit [Akin81] bleek dat bij een bepaald electronic mail systeem gewone gebruikers veel meer fouten maken dan geoefende gebruikers, en daarom veel vaker dokumentatie moeten raadplegen. Het dokumentatie systeem dient daarom efficient te zijn. Overigens kan automatische foutenkorrektie hier ook helpen.

Veel systemen hebben tegenwoordig help funkties die uitleg geven over het gebruik van bepaalde faciliteiten (bijv. de syntax van de kommando's).

---

*) Wanneer men hierin slaagt komt een belangrijk probleem aan de orde: wie zal de "wereldvisie" van deze computersystemen genereren? Een dergelijk systeem zou grote mogelijkheden voor (politieke en sociale) manipulatie kunnen bieden. Of zou hier ook een verzuiling kunnen optreden?

Veel minder ziet men de omgekeerde informatie: welke kommando's men nodig heeft om een bepaalde aktie uit te voeren. Een goede help funktie heeft ook de mogelijkheid om zich aan te passen aan de kontext waarin de gebruiker zich bevindt, en aan het expertise niveau van de gebruiker.

3.0 Raakvlakken.

Zoals reeds eerder is opgemerkt, zijn er diverse raakvlakken tussen de verschillende genoemde gebieden, zowel wat betreft de verschillende interaktieve toepassingen als de interaktiemethoden. In de huidige te koop zijnde systemen heeft men meestal te maken met onafhankelijk van elkaar werkende komponenten (subsystemen) en moet een overgang gemaakt worden (via het operating systeem) tussen de komponenten wil men van soort werk veranderen. Nog moeilijker wordt het als men gegevens van één terrein wil gebruiken in een ander, of methoden die op één gebied voorhanden zijn wil toepassen op een ander. In dat geval is men meestal aangewezen op konversieprogrammatuur (indien al mogelijk), waardoor men volkomen in de genoemde semi-interaktieve werkwijze terecht komt. In deze sektie worden suggesties gegeven van verschillende mogelijkheden tot integratie op deze raakvlakken.

3.1 Database met tekstverwerking.

Er zijn hier 6 toepassingen waar database en tekstverwerkingstechnologieen elkaar raken:

. Het invoegen van queries in een tekst.

Om dit te kunnen doen, is bij de meeste systemen een aantal stappen nodig, nl. 1) het doen van de query m.b.v. een database systeem; 2) het opbergen van het resultaat van de query op een file; 3) meestal konversie van deze file naar een tekstverwerker formaat; 4) in de tekstverwerker de plaats voor het query resultaat opzoeken; 5) het invoegen van de query resultaat file; 6) eventueel herformatteren van het query resultaat. De werkwijze is nogal omslachtig en, erger, meestal moeilijk of niet te automatiseren, d.w.z. een groot deel van deze handelingen moet telkens handmatig uitgevoerd worden. Om dit soort werkzaamheden vlot te laten verlopen (bijv. voor regelmatig terugkerende rapporten), moet het mistens mogelijk zijn om de query (of een parametriseerbaar skelet ervan) in het tekstdokument op te slaan, evenals eventueel benodigde formatteringsaanwijzingen. In dit geval krijgt de tekstverwerker ook de

kenmerken van een report writer. [Kais80] beschrijft een tekstformatter, waar ook SQL queries tussen de tekst gegeven kunnen worden.

. Het formatteren van query resultaten.

In feite hierboven al genoemd. Dit wordt meestal aan een z.g. report writer overgelaten. Aangezien formattering een tekstverwerkings funktie is, kunnen we de principes die hier in gebruik zijn toepassen op de specifieke database strukturen. Anderzijds is het zo dat database strukturen andere formatteringseisen kennen (kolom-rangschikking, sortering, groepering), die ook in algemene tekstdokumenten nuttig kunnen zijn.

. Het parametriseren van tekst.

Een van de faciliteiten die tekstverwerkers vaak bieden, is het maken van gestandaardiseerde brieven, waarbij enkele variabelen uit een database kunnen worden ingevuld. Hiervoor moet dan een selektie uit de database gemaakt worden. De selektiemogelijkheden zijn i.h.a. beperkt in vergelijking met een goede query-taal. In feite kan men deze mogelijkheid beschouwen als een formattering van een query-rapport, waarbij per record een dokument aangemaakt wordt. Soms zal het nodig zijn om eerst groepering toe te passen en per groep een dokument aan te maken (bijv. wanneer men een brief per organisatie stuurt, en de namen van alle betrokken personen in de brief noemt). In dat geval komt men op dezelfde problemen als het invoegen van query-resultaten in een tekstdokument.
Bij deze toepassing zijn er ook duidelijke verbindingslijnen naar electronic mail.

. Het organiseren van dokumenten.

Wanneer men een omvangrijk bestand heeft van dokumenten, zoals dat bij uitgebreide kantoorautomatisering te verwachten is, zal het nodig zijn om dokumenten te organiseren in een struktuur, zoals men dat met andere data doet in een database. Hierbij moet men denken aan verschillende sleutels, zoals datum, auteur, geadresseerde(n), onderwerp, e.d., terwijl ook de relaties tussen verschillende dokumenten moet kunnen worden weergegeven (zoals revisie van, antwoord op, overzicht van, etc.). Vooral als ook electronic mail meegenomen wordt in het systeem, moeten opbergmogelijkheden zoals in een kantooromgeving gebruikelijk zijn, geboden kunnen worden. Vanwege de uniformiteit is het gewenst, om hiervoor niet een apart systeem te hebben, of een eigen funktie van de tekstverwerker, maar om de volledige database faciliteiten hier te kunnen toepassen.

. Databases met tekst.

De meeste database systemen hebben alleen mogelijkheden voor data met een starre indeling (numeriek, vaste lengte strings), waardoor meer

flexibele data, zoals een literatuurverzameling, moeilijker te hanteren zijn. Bij deze soort data willen we graag zo mogelijk formatteringsgegevens in de database opslaan, en er zijn vaak een variabel aantal sleutels per record mogelijk (bijv. onderwerpsrubrieken).

. Indexen in een dokument.

Sommige tekstverwerkers hebben voorzieningen om van een dokument een index te genereren, waarin woorden gezet worden, die in het dokument aangegeven worden. Deze indices zijn handig voor de lezer van (de afdruk van) het dokument om iets op te zoeken. Bij het zoeken van woorden met de tekstverwerker moet echter vaak het hele dokument sequentieel gelezen worden. Voor dokumenten die veel gebruikt worden als naslagwerk, kan een index, zoals gebruikelijk in database systemen een aanzienlijke versnelling van de zoektijd geven.

3.2 Tekstverwerking met rekenmodellen.

In tekstdokumenten komen nogal eens numerieke overzichten voor. Wanneer hierbij berekeningen uitgevoerd moeten worden (bijv. totalen) dan bespaart het werk wanneer deze door de tekstverwerker worden uitgevoerd, d.w.z. als de funkties van een elektronisch kladblok in de tekstverwerker geintegreerd zijn. Een elektronisch kladblok biedt op zich meestal te weinig funktionaliteit omdat het niet de totale tekstverwerkingsmogelijkheden biedt, die nodig zijn om het dokument te voltooien (formattering van begeleidende tekst, layout-verzorging van het rekenmodel). Sommige tekstverwerkers hebben een aantal mogelijkheden van het elektronisch kladblok ingebouwd. Als belangrijk alternatief voor het maken van berekeningen moet nog genoemd worden het kontroleren van berekeningen. Door de aldus toegevoegde data-overtolligheid kunnen fouten in cijfermateriaal ontdekt worden.

3.3 Database met rekenmodellen.

De invoergegevens voor een rekenmodel zullen vaak uit een database gehaald kunnen worden. Bij de huidige systemen zal de gebruiker dan vaak van verschillende programma's gebruik moeten maken met mogelijk nog een konversie ertussen. Ook het terugbergen van gewijzigde gegevens van een rekenmodel in de database zou vereenvoudigd kunnen worden als de funkties van een database en elektronisch kladblok geintegreerd kunnen worden. Hiertoe zou men bijv. een query als onderdeel van een rekenmodel moeten

kunnen definieren.

3.4 Formulieren met tekst.

Formulieren kunnen beschouwd worden als een speciale vorm van tekstverwerkingsdokumenten, nl. met een nogal starre struktuur. Anderzijds kan er behoefte bestaan aan formulieren met wat meer struktuurvrijheid, bijvoorbeeld met (flexibele) vakjes voor begeleidende tekst, variabel aantal rijen van een bepaald soort, e.d. waardoor de verworvenheden van tekstverwerking hier vruchtbaar toegepast kunnen worden. De boven reeds genoemde standaardbrieven kunnen beschouwd worden als een dergelijk formulier, waarbij de basis een tekstverwerkingsdokument is, en de parameters overeenkomen met de in te vullen vakjes van een formulier.

3.5 Formulieren met rekenmodellen.

Op formulieren komen vaak eenvoudige berekeningen voor. Zo zal op een bestelformulier de berekening "prijs = aantal * stuksprijs" voorkomen met daarnaast totaalberekeningen, belastingberekeningen e.d. Wanneer een informatiesysteem de mogelijkheid biedt om deze berekeningen als onderdeel van een formulier te definieren, dan is er geen (gebruikers-) programmatuur nodig om deze triviale bezigheden uit te voeren.

3.6 Formulieren met databases.

Op een formulier komen vaak gegevens voor die uit een database gehaald worden. Bij het hierboven genoemde voorbeeld van een bestelformulier zal de stuksprijs uit een database gehaald worden met als sleutel een artikelnummer, terwijl bijv. een kortingspercentage als funktie van een klantidentifikatie uit een andere database beschikbaar kan zijn. Wanneer we deze mogelijkheid hebben, kunnen we een formulier gebruiken als twee-dimensionale query-interface. De velden op het formulier waar een sleutel ingevoerd moet worden funktioneren dan als "trigger" voor een operatie op de database. Samen met de genoemde mogelijkheid van berekeningen in een formulier levert dit een krachtig hulpmiddel om administratieve toepassingen te bouwen (vergelijk SBA, [Zloof77b]).

4.0 Informatiestrukturen en operatoren.

In het voorgaande zijn verschillende onderdelen van een informatiesysteem bekeken n.a.v. de verschillende funktionele mogelijkheden. Daarbij kwamen terloops ook de verschillende strukturen ter sprake, die we kunnen gebruiken om informatie te organiseren. In deze sektie zullen we onderzoeken welke strukturen gebruikelijk zijn in de verschillende subsystemen en van daar uit een voorstel formuleren voor een model dat deze strukturen omvat.

4.1 Informatiestrukturen.

Om te beginnen beschouwen we een simpele administratieve toepassing, bijv. een kollegeadministratie. Hierbij hebben we een bestand, waarin gegevens over studenten zijn opgeslagen (naam, adres, woonplaats, jaar van aankomst, studierichting, etc.) en bij elke student gegevens over de kolleges die hij volgt (gevolgd heeft) en de tentamen- of praktikumresultaten. Bij een handmatige administratie hebben we een aantal mogelijkheden om deze informatie bij te houden: de meest gebruikte zijn lijsten (een regel per student) en systeemkaarten (een kaart per student). Het laatste wordt vooral gebruikt, waar per student een variabel aantal gegevens nodig is. In termen van database modellen komt de lijst overeen met een relatie (relationele model), en de kaart met de set (DBTG model) of de ouder-kind relatie (hierarchisch model). Beide kunnen op een beeldscherm voorgesteld worden door een (flexibel) formulier, i.h.b. de systeemkaart. Wijzigingen op de kaart kunnen dan op dit formulier uitgevoerd worden, zoals men met een tekstverwerker werkt. Zoals een scherm editor voordelen blijkt te hebben boven een lijn editor, zou een "full-screen" database interface voordelen kunnen hebben boven de gangbare query-talen. De ervaringen met QBE wijzen in deze richting. Vergelijkend onderzoek zal moeten uitwijzen of dit inderdaad het geval is.

Verschillende tekstverwerkers geven de mogelijkheid om data in kolommen te organiseren, waardoor we een voorstelling van een relatie in het relationele model benaderen. Een dergelijke tekstverwerker kunnen we gebruiken om een primitieve administratie bij te houden. Om de gegevens van een bepaalde student Jansen te vinden, kunnen we het zoekkommando van de editor gebruiken. Helaas zullen we op deze manier soms i.p.v. een student Jansen iemand vinden die in een Jansenstraat woont, of iets dergelijk. Het probleem is dat de editor te weinig struktuur aan het dokument toekent. We zouden graag willen zoeken in een bepaalde kolom. Bovendien is er behoefte aan meer geavanceerde operatoren, zoals het zoeken op intervals van waarden (range queries), het maken van selekties en projekties, het definieren van

datatypen, e.d.

Teksteditors houden zich voornamelijk bezig met de uiterlijke verschijningsvorm van data, en veel minder met de interne struktuur, terwijl database systemen juist het tegenovergestelde doen. Wanneer we ons gaan bezig houden met integratie van deze systemen is het belangrijk het onderscheid te blijven zien tussen de interne struktuur van een informatiesysteem en de uiterlijke verschijningsvorm (externe struktuur). We zullen in het volgende beide aspekten van verschillende subsystemen aangeven.

4.1.1 Teksteditors. - Een eenvoudige teksteditor kent als struktuur alleen de regelindeling (twee-dimensionale layout). Een tekstverwerker kent daarnaast i.h.a. nog een alinea- en paginaindeling en eventueel nog een kolommenstruktuur. Deze struktuur is voornamelijk gebaseerd op de layout van het dokument. Wat ontbreekt is een uitgebreidere struktuur die gebaseerd is op de relaties tussen de data. Als voorbeelden hiervan kunnen genoemd worden de indeling in hoofdstukken, sekties, e.d, voetnoten, en verwijzingen tussen de tekst (o.a. index en inhoudsopgave). Sommige van deze strukturen zijn meestal ad hoc aanwezig, zonder dat sprake is van een omvattend conceptueel geheel. Een belangrijke taak voor het wetenschappelijk onderzoek is het maken van dergelijke concepten.
Een interessante aanpak om tot een omvattend model te komen voor layoutstrukturen vindt men in [Cham81]. De tekst wordt hierbij geacht te worden geplaatst in dozen van verschillend formaat, met enige flexibiliteit. Een normale tekst staat in een doos die aan de onderkant langer kan worden, een voetnoot in een doos, die naar boven kan groeien. Bij deze dozen zijn dan akties gedefinieerd, die uitgevoerd worden als er een "botsing" met een andere doos plaatsvindt. Op deze manier kunnen diverse ad-hoc technieken in een overzichtelijk raamwerk worden geplaatst.

Als interne struktuur van tekstdokumenten kunnen we nemen de zojuist genoemde hoofdstuk-, alinea- en sektieindelingen, en de verwijzingen tussen tekst. Als generalisatie van voetnoten kunnen ook in aanmerking genomen worden voorzieningen om tussen de tekst opmerkingen, aantekeningen e.d. op te nemen, die wel bij een bepaalde plaats in de tekst horen, maar toch logisch ervan onderscheiden zijn. Hierbij zouden niveaus van belangrijkheid toegevoegd kunnen worden. Het idee om tekst op deze manier te struktureren is uitgewerkt in diverse struktuureditors (zie 3.2).

De externe struktuur van een tekstdokument is de layout. Deze hangt natuurlijk samen met de interne struktuur van het dokument. De layout zal gestuurd worden door de indeling in hoofdstukken, door het voorkomen van voetnoten e.d. Toch is het mogelijk om binnen bepaalde grenzen de layout te wijzigen, zonder dat de interne struktuur of de inhoud van het dokument

wijzigt. Enkele voorbeelden hiervan zijn: de pagina grootte kan onafhankelijk gewijzigd worden, dubbele spatiering kan gekozen worden, voetnoten kunnen naar keuze per pagina, per hoofdstuk of aan het eind van het dokument gegeven worden. Verder kan voor bepaalde konstrukties het lettertype gewijzigd worden: onderstrepingen kunnen vervangen worden door kursief, e.d. Wanneer we verschillende niveaus van annotaties onderscheiden kunnen we kiezen welke wel en welke niet in de output opgenomen worden. Deze mogelijkheden hebben we niet alleen bij het afdrukken van een dokument, maar ook tijdens het werken met de tekstverwerker. Tijdens een editoperatie moet de gebruiker dan ook op vrij eenvoudige wijze kunnen kiezen of bepaalde gedeelten van het dokument al dan niet getoond moeten worden op het beeldscherm, resp. of een bepaalde externe struktuur zichtbaar gemaakt moet worden. In de meest extreme vorm hiervan kan de gebruiker kiezen tussen het zien van het ruwe dokument, inklusief formatteringsvoorschriften, struktuuraanwijzingen, etc. enerzijds, en het werken met het uiteindelijke resultaat van de formattering anderzijds.

4.1.2 Databases. - Bij een database bestaat de interne struktuur uit de definitie van het te gebruiken datamodel, de datatypen, en de onderlinge relaties tussen de data-elementen. Welke mogelijkheden hier aanwezig zijn, hangt af van de het datamodel dat gekozen wordt. Behalve de reeds genoemde operatoren selektie en projektie is hier ook belangrijk de join operator, waarmee verschillende delen van een informatiesysteem gekoppeld worden. Deze operator is, zoals eerder opgemerkt is, in het algemeen voor gebruikers conceptueel vrij moeilijk, hoewel in de praktijk er veel gebruik van gemaakt wordt. Ook bij handmatige administraties komt deze operatie veel voor, nl. als informatie van diverse bronnen bijeengevoegd wordt op grond van een gemeenschappelijk element (sleutel). Als voorbeeld kunnen we nemen het samenstellen van tentamenbriefjes, waarbij informatie uit het tentamenbestand (cijfer, docent), moet samengevoegd worden met informatie uit het studentenbestand (naam, studierichting, evt. adres). In feite is het maken van een verzameling tentamenbriefjes, evenals het maken van standaardbrieven ook een join-operatie, nl. op een database selektie als de ene operand en een tekstdokument (of formulier) als de andere operand. Het verdient aanbeveling om te experimenteren met verschillende mogelijkheden om join-operaties op natuurlijke wijze voor te stellen.

De externe struktuur van een database omvat zowel de representatie van de datatypen, als de representatie van relaties. Wat de datatypen betreft, hebben we bij numerieke waarden verschillende formatteringsmogelijkheden. Relaties tussen dataelementen kunnen weergegeven worden d.m.v. tabellen, met of zonder groepering, boomstrukturen, e.d. Ook het voorstellen van records van een database of groepen ervan op een formulier kan als een vorm van externe struktuur beschouwd worden. In dit opzicht is het genereren van een verzameling tentamenbriefjes vanuit een database te beschouwen als een

definitie van een bepaalde externe struktuur (als funktie van de interne), zonder dat hierbij sprake is van echte database operaties. Het sorteren van een database overzicht op verschillende sleutels kan op dezelfde manier als een externe struktuur operatie beschouwd worden. Op deze manier is het dus mogelijk om de externe struktuur te wijzigen zonder dat de interne struktuur aangepast wordt. Omgekeerd kan men de interne struktuur wijzigen, of de data aanpassen, zonder dat men zich zorgen hoeft te maken over de externe struktuur, omdat de afbeeldingsfunkties hetzelfde blijven.

4.1.3 Elektronisch kladblok. - Bij een elektronisch kladblok bestaat de interne struktuur uit de definitie van de matrix, de datatypen van de elementen, en de funkties die de relaties tussen de elementen beschrijven. De externe struktuur bestaat uit de representatie van de elementen op het scherm. Bij deze toepassing is het verschil tussen interne en externe struktuur duidelijk geillustreerd: de formules, die de struktuur beschrijven worden meestal onzichtbaar gehouden, terwijl het resultaat ervan getoond wordt. Anderzijds is het soms noodzakelijk om de onderliggende formules te zien, bijv wanneer het model opgezet of gewijzigd wordt. In dat geval kiest de gebruiker tijdelijk voor een andere externe representatie. De externe struktuur kan ook gewijzigd worden door bijv. bepaalde kolommen niet te laten zien, of door een pagina-indeling aan te brengen.

4.1.4 Formulieren. - Voor formulieren geldt bijna hetzelfde als voor het elektronisch kladblok. Als onderdeel van de externe struktuur zouden we hier eventueel kunnen beschouwen de vaste tekst (i.e. de formulierindeling), terwijl de diverse attributen van de velden (numeriek, moet ingevuld worden, e.d. als onderdeel van de interne struktuur kunnen beschouwen. Er is reeds op gewezen dat bij een formulier er een onderscheid gemaakt moet worden tussen de editor voor de layout en attributen en voor het editwerk bij het invullen van het scherm. We kunnen nu stellen dat we hier niet hoeven te spreken over verschillende editors, maar dat we een andere externe struktuur toepassen (nl. een die de interne struktuurelementen zichtbaar maakt).

4.2 Struktuureditors.

Reeds lang is de behoefte gevoeld aan editors die in staat zijn om meer interne strukturering in een dokument aan te brengen. Een vroeg voorbeeld van zo'n struktuureditor is het NLS Hypertextsysteem, zie o.a. het klassieke overzicht van Van Dam en Rice [VDam71]. Deze editor gaf de mogelijkheid om een boomstruktuur in de tekst aan te brengen. De laatste tijd is er een

hernieuwde belangstelling voor het onderzoek naar struktuureditors en de mogelijke uitbouw naar meer algemene kantoorinformatiesystemen.

De reeds genoemde scherm editor van Irons en Djorup [Iron72] werd uitgebouwd tot een algemene editor waarmee het mogelijk is edit operaties op datastrukturen uit te voeren (bijv. bomen, lijsten) [Fras80, Fras81a, Fras81b].

In Zuerich wordt gewerkt aan een familie van struktuurgeorienteerde editoren voor de personal computer Lilith, waarmee op soortgelijke wijze een boomstruktuur in dokumenten aangebracht kan worden [Burk80, Burk81]. Deze editors hebben ook meer algemene mogelijkheden, zoals bijvoorbeeld syntax-sturing (voor interaktief ontwikkelen van programma's).

Aan de universiteit van Linkoeping wordt het Linkoeping Office Information System (LOIS) ontwikkeld als generalisering van een struktuurgeorienteerde editor [Strom81]. Aan het MIT denkt men met het Etude systeem in dezelfde richting [Hamm81].

Bij deze systemen gaat men i.h.a. uit van een bepaalde editor (of krachtiger tekstverwerker), waar men extra struktureringsmogelijkheden aan toevoegt. Zij zijn in eerste instantie sterk op tekststrukturen georienteerd. Het is daarom de vraag in hoeverre het mogelijk is om met deze ontwikkelingen op adekwate wijze toegang tot algemene database systemen te verkrijgen. En vooral, hoe op natuurlijke wijze de gebruiker een eigen informatiesysteem te laten opzetten. Het is niet ongerechtvaardigd, te proberen ook onderzoek op dit gebied op te starten vanuit de richting van databases, en de daar aanwezige concepten te laten inwerken op de editor ontwikkelingen.

4.3 Operatoren.

We hebben gezien dat de externe struktuur maar tot op zekere hoogte vrij te kiezen is. Beter is het daarom te spreken over de afbeelding van interne naar externe struktuur. De gebruiker heeft een aantal operatoren tot zijn beschikking, waarmee de externe struktuur als funktie van de interne struktuur gedefinieerd wordt.

Aangezien de externe struktuur van een dokument (file) in het algemeen veel groter is dan wat op een beeldscherm zichtbaar gemaakt kan worden, moeten we het beeldscherm beschouwen als een venster op de externe struktuur. Het is natuurlijk mogelijk dit venster kleiner te maken dan het beeldscherm. Dit geeft dan ook de mogelijkheid om op een beeldscherm meerdere vensters tegelijk te hebben, die verschillende delen van een

dokument laten zien, of delen van verschillende dokumenten. Dit kan het werken met informatie uit verschillende bronnen aanzienlijk vergemakkelijken. Ook het werken met menu's kan hierdoor eenvoudiger zijn omdat het menu een venster in beslag kan nemen. Smalltalk is een goed voorbeeld van zo'n venstersysteem. De gebruiker moet daarom naast de operaties voor het manipuleren van interne en externe struktuur, ook operaties voor het manipuleren van de vensters op het scherm hebben. Belangrijke operaties hiervoor zijn: het definieren van een venster, vergroten en verkleinen, verschuiven, en het "naar voren halen" van een venster (als vensters over elkaar heen liggen).

Voor een goed bruikbaar systeem dienen we te zoeken naar operatoren die de gebruiker ter beschikking staan om zowel de data, als de interne struktuur, de relatie interne-externe struktuur en de vensters te manipuleren. Hierdoor kan de gebruiker volstaan met een minimale hoeveelheid concepten, die in al deze situaties bruikbaar zijn. Als voorbeeld geven we een vergelijking van data-, interne en externe operaties voor een aantal systemen.

Tekstverwerking.
Een basisoperatie is het selekteren van een stuk tekst. We kunnen nu dit geselekteerde stuk:
- weglaten of verplaatsen (data operatie).
- in de achtergrond zetten (interne struktuur).
- van een ander lettertype voorzien (externe funktie).

Database.
Als basisoperatie nemen we de selektie van een record of kolom.
- breng een wijziging aan (data operatie).
- verplaats het record naar een andere DBTG set (intern).
- laat de kolom weg van het scherm, maar niet uit de database (extern).

Elektronisch kladblok.
Als basisoperatie nemen we weer de selektie van een element.
- wijzig het element (data operatie).
- wijzig de formule voor dit element (intern).
- wijzig de layout van het element, bijv. het aantal decimalen (extern).

4.4 Een geintegreerd systeem.

Een geintegreerd systeem biedt de gebruiker een orthogonale verzameling elementaire strukturen en operatoren, waarmee hij zijn informatiesysteem kan opbouwen. De strukturen die het systeem moet bieden zijn de elementaire vormen die we in het voorgaande bij de verschillende subsystemen zijn

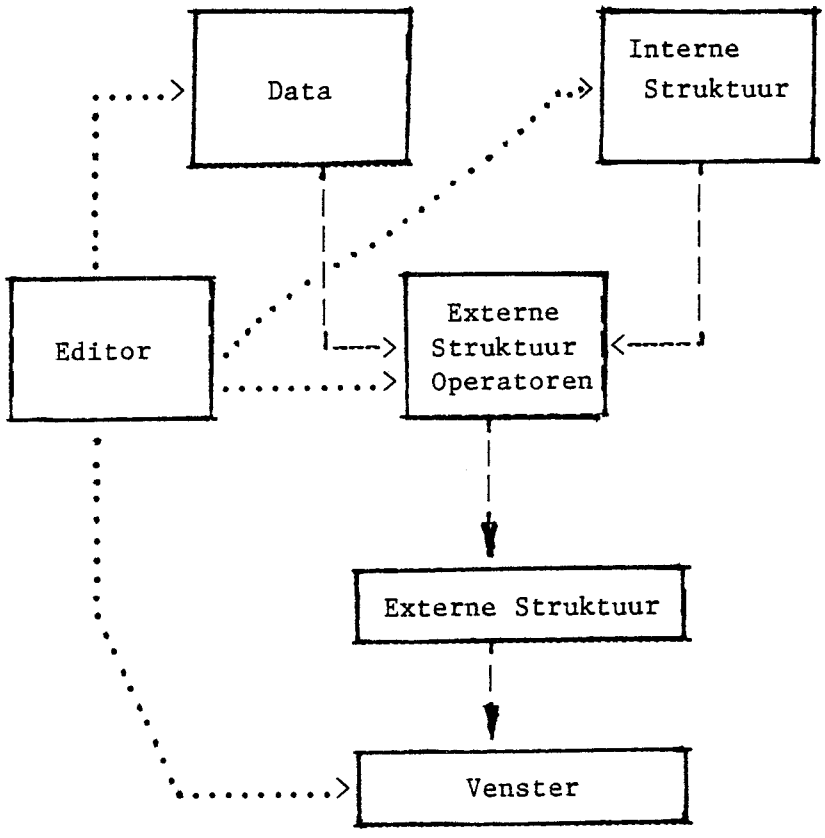
tegengekomen: tekststrukturen, relaties, boomstrukturen, e.d. Deze strukturen zijn niet gebonden aan een bepaalde toepassing: een relatiestruktuur kan zowel voor een "database" als voor een "formulier" worden gebruikt, etc. Ook binnen een bestand kunnen gedeelten voorkomen met verschillende strukturen. De gebruiker kan deze elementaire strukturen gebruiken om zijn informatiestrukturen op te bouwen: tekstdokumenten, databases, formulieren, menu´s, rekenmodellen en combinaties ervan. Bovendien moeten dezelfde strukturen en operatoren beschikbaar zijn om "directories" op te bouwen.

De operatoren die ter beschikking staan kunnen gebruikt worden om de data te manipuleren, de interne strukturen op te zetten en te wijzigen, de relatie interne-externe struktuur te wijzigen, en de vensters te manipuleren. Voorbeelden van operatoren zijn: toevoegen, weglaten en veranderen, selekteren, opzoeken. Onderzoek is nodig om een goede kollectie operatoren te vinden, die conceptueel eenvoudig is, en toch een grote kracht heeft. Verder moeten mechanismen gevonden worden om de operatoren in de juiste kontext toe te passen (bijv. om te voorkomen dat een operatie die bedoeld is voor de interne struktuur, op de data toegepast wordt.

Een van de belangrijkste eigenschappen van een goed systeem is de mogelijkheid om "procedures" ("macro´s") van veel voorkomende operaties vast te leggen. Tenslotte is de kracht van computers o.a. gelegen in het feit dat ze eindeloos bepaalde herhalingen kunnen uitvoeren. De gebruiker moet zo kunnen programmeren. Hierbij komen alle problemen die bij het programmeren kunnen ontstaan, ook aan de orde. In het bijzonder de keuze en representatie van besturingsstrukturen, en het weergeven van variabelen zijn zaken die een grondige studie vereisen. QBE is een voorbeeld van een mogelijke aanpak. Het probleem van de besturingsstrukturen is hierbij (en bij andere relationele database systemen) iets afgezwakt omdat de operatoren op verzamelingen records (tuples) werken i.p.v. op enkele records.

De te definieren macro´s of procedures zullen weer een onderdeel vormen van de interne struktuur van een dokument. Als zodanig staan de struktureringsmogelijkheden en de operaties die op dokumenten uitgevoerd kunnen worden ook hier ter beschikking. Interessante mogelijkheden zijn hierbij om procedures te koppelen aan operaties die plaatsvinden op te specificeren gedeelten van een dokument ("triggers"). Voorbeelden hiervan zijn: een veld op een formulier kan een trigger hebben die zorgt dat andere velden uit een database gehaald worden wanneer hier iets ingevuld wordt; een wijziging in een directory kan tot gevolg hebben dat een ander dokument wijzigt of verdwijnt. In het TLA systeem is een soortgelijke voorziening opgenomen, genaamd "smart forms" [Tsich82].

Het geintegreerde systeem kunnen we a.v. voorstellen (fig.1):



....> edit operaties ------> dataflow

fig. 1.

Op deze manier zou een op zichzelf staand systeem opgezet kunnen worden. De behoefte aan "echt" programmeerwerk zal hierdoor echter niet verdwijnen. Vooral voor kant-en-klare toepassingspakketten zal een grote markt blijven bestaan. Deze, en ad-hoc te maken programma's hebben vaak behoefte aan interaktievoorzieningen zoals in dit rapport beschreven. Er is voor verschillende computersystemen formulier- en menubeheerssoftware te koop waarmee op eenvoudige wijze goede interaktie in het programma ingebouwd kan worden. Het hier voorgestelde systeem kan als een uitbouw hiervan beschouwd worden, en zou aan toepassingsprogrammatuur standaard edit- en schermbeheersfunkties kunnen bieden. Het systeem zou dan ongeveer als volgt gestruktureerd zijn (fig.2):

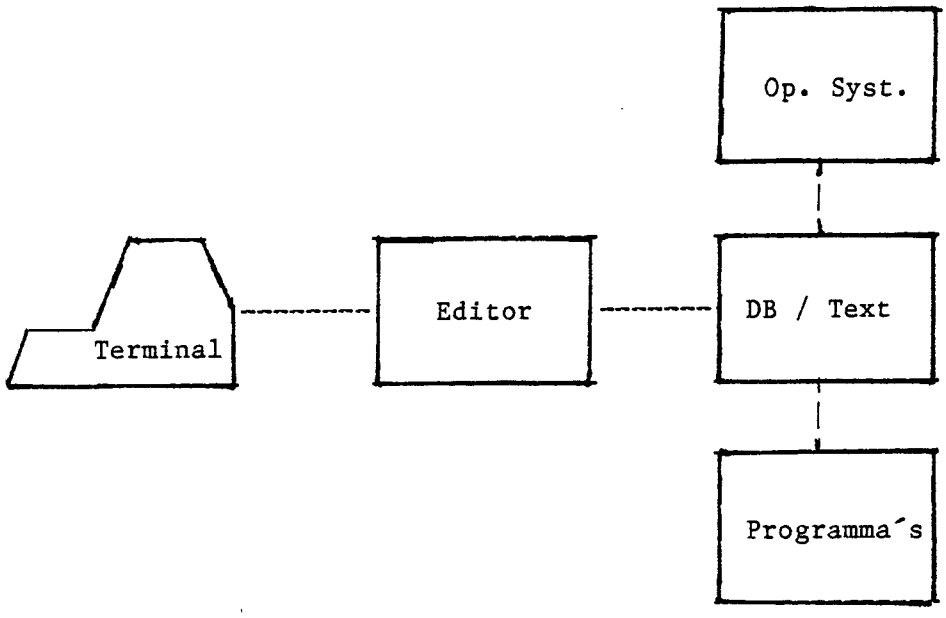


fig. 2.

In een dergelijke opzet kan de editor als onderdeel van het operating systeem beschouwd worden (terminal driver). In de terminal drivers van de moderne operating systems zitten al faciliteiten voor eenvoudige scherm operaties, zoals het weghalen van een karakter wanneer de RUBOUT toets wordt ingedrukt, het adresseren van een cursor, e.d. Het voordeel van een dergelijke voorziening in een O.S. is dat meer programmatuur gebruik zal maken van goede interaktieve voorzieningen, dat de uniformiteit toeneemt, en dat de benodigde software slechts een maal in het systeem aanwezig is. Wanneer dit wordt toegepast op een groter systeem met meerdere gebruikers, dan kan een dergelijke voorziening een betere prestatie opleveren, wanneer de editfunkties op een laag niveau in het systeem worden uitgevoerd, omdat dan minder swapping nodig is. Wanneer deze funkties in een applicatieprogramma worden uitgevoerd is vaak per ingetypt symbool een aktie van het programma nodig om het scherm aan te passen e.d. Bovendien is het zo mogelijk om de intelligentie voor deze operaties te distribueren (bijv. gedeeltelijk in een slimme terminal).

Verschillende onderzoekers hebben de verwachting uitgesproken dat edit funkties een belangrijk aandeel in toekomstige systemen zullen hebben. Burkhart merkt op dat editors naar zijn verwachting meer en meer een belangrijke rol als algemene interface zullen gaan spelen en daarom belangrijke researchobjecten voor dit decennium zullen zijn [Burk81]. Soortgelijke verwachtingen zijn uitgesproken door Elliot [Elli82].

5.0 Scenario.

De studie van geintegreerde informatiesystemen heeft veel aspekten. Ook zijn er verschillende raakvlakken met andere delen van de informatica. We noemen hierbij in het kort de belangrijkste studie onderwerpen:

. Prototype.
Om ervaring op te doen met de strukturen en operatoren die mogelijk zijn, is het nodig een prototype voor een eenvoudig subsysteem te maken. Gezien de wenselijkheid van de databases als uitgangspunt, die reeds eerder is geformuleerd, is een goede toepassing hiervoor een systeem waarmee eenvoudige databases gebouwd kunnen worden (bijv. gebaseerd op het relationele systeem.

. Basis strukturen.
Onderzoek naar de gewenste strukturen waarmee de verschillende gewenste subsystemen gerealiseerd kunnen worden. De strukturen moeten overeenkomen met een conceptueel model wat de gebruiker in zijn hoofd heeft. D.w.z. als de gebruiker denkt aan lijsten, tabellen, formulieren e.d. dan moeten deze ook als datastrukturen gebruikt kunnen worden.

. Atomaire operaties.
Hiervoor geldt hetzefde als voor de basis strukturen. De operatoren moeten simpel zijn, orthogonaal en toepasbaar voor de verschillende aspekten (data, struktuur, etc.). Bovendien moet worden nagegaan hoe de operatoren door de gebruiker gegeven moeten worden (menu, funktietoetsen, e.d), hoe de gebruiker zelf operatoren kan definieren etc.

. Programma interfaces.
Welke voorzieningen kunnen aan programma's geboden worden om interaktieve toepassingen op elegante en efficiente wijze mogelijk te maken?

. Externe struktuur operatoren.
Welke operatoren zijn er nodig om de gewenste funkties van interne naar externe struktuur te kunnen definieren. Wat zijn de defaults die het systeem moet geven?

. Macro faciliteiten.
Hoe moet de gebruiker macro's (procedures) specificeren? Een van de methoden is dat de gebruiker het systeem "voordoet" wat de bedoeling is ("Programming by example"). Hoe geeft de gebruiker dan aan wat een variabele is, hoe worden verschillende besturings strukturen weergegeven, e.d.? Vooral konditionele operaties komen nogal eens voor in informatiesystemen. Hiervoor zou bijv. een systeem met

beslissingstabellen handig kunnen zijn, omdat dit nauw aansluit bij de andere aanwezige concepten.

- Query optimalisatie.
  Voor toegang tot grote databases (i.e. dokumenten met een bijzondere toegangsstruktuur zoals een index) kan query optimalizatie toegepast worden. Het onderzoek naar relationele databases heeft hiervoor al veel resultaten beschikbaar, maar deze zijn meestal gebaseerd op het feit dat de gehele query van te voren beschikbaar is en dat het gehele resultaat van de query afgeleverd moet worden. Bij een systeem, waarin de gebruiker de query stap voor stap opbouwt uit onderdelen, kan de optimalisatieprocedure ook alleen maar incrementeel toegepast worden. Bovendien hoeft een query niet altijd het gehele resultaat af te leveren: datgene wat niet op het venster zichtbaar wordt behoeft in principe nog niet opgezocht te worden. Zo zou het mogelijk zijn om tijdens het uitvoeren van een query op een andere methode over te gaan. Het is duidelijk dat hier nog veel onderzoek verricht kan worden.

- Database Modellen.
  Welke database modellen zijn geschikt om in een geintegreerd informatiesysteem ingebouwd te worden, en hoe kunnen de beschikbare strukturen gebruikt worden om deze te realiseren? Ook van belang hierbij is de vraag hoe een geintegreerde interface gekoppeld kan worden met bestaande file- en database management systemen.

- Triggers.
  Wat voor automatische akties kunnen we standaard voorzien en hoe worden deze gekoppeld aan de informatiestrukturen?

- Beveiliging.
  Beveiligingsaspekten zijn: bescherming van de data tegen korruptie: automatische backupmogelijkheden, bescherming tegen ongeautoriseerde toegang, bescherming bij gelijktijdige toegang (bij multi-user systemen). In hoeverre zijn er verschillen met standaard systemen, waar de informatiestrukturen homogener zijn en de operatoren minder atomair.

- Netwerken.
  Wat zijn de konsekwenties voor het bouwen van netwerken van informatiesystemen? Hierbij moeten we ook denken aan de toepassing in kantoorautomatisering. Daarvoor is niet alleen de interface met de enkele gebruiker van belang maar vooral ook de mogelijkheid om kantoorprocedures te kunnen specificeren, die o.a. aangeven hoe de informatie door een kantoor gerouteerd wordt [Tsich82, Zloof77b]. Verder kunnen elementen van de software gedistribueerd worden over verschillende subsystemen (terminal servers, file servers, e.d.). Hier is nog veel ontwikkeling mogelijk op het gebied van protokollen, die een efficiente gedistribueerde verwerking mogelijk maken.

. Hardware.

Welke hardware is nodig voor een geintegreerd informatiesysteem en welke is gewenst (grafische I/O, muis, e.d)? Welke speciale computerarchitektuur komt in aanmerking (tekstverwerking, database machines, etc.)?

. Software koppelingen.

Wat voor mogelijkheden zijn er om software komponenten te koppelen en om bestaande software in te bouwen? Het Unix systeem biedt mogelijkheden om software komponenten achter elkaar te schakelen, maar dit is gebaseerd op een sequentiele behandeling van informatie. Is iets dergelijks ook mogelijk bij een scherm-georienteerd informatiesysteem, en zo ja, wat voor primitieven moet het systeem dan bieden om dit te vergemakkelijken?

. Kwantitatieve en kwalitatieve evaluatie.

Van alle strukturen, operatoren, e.d. die bedacht worden zal nagegaan moeten worden, wat hun effektiviteit is, zowel m.b.t. het gebruikersgemak als m.b.t. de efficientie van de implementatie. Hiervoor moeten in de prototypes meetpunten en tellingen ingebouwd worden. Voor het gebruikersaspekt is daarbij nodig dat er psychologische experimenten uitgevoerd worden.

6.0 Besluit.

Het lijkt mogelijk om een belangrijk deel van het werken met informatie m.b.v. een computer te integreren in een uniform interaktief systeem. Hiermee is niet gezegd dat alle interaktieve werkzaamheden door één systeem uitgevoerd kunnen worden. Het is echter aan te bevelen voor zoveel mogelijk werkzaamheden een eenduidige interface te gebruiken.

De principes die in interaktieve systemen in gebruik zijn, blijken in veel opzichten dichter bij elkaar aan te sluiten dan men op het eerste gezicht verwacht. Bovendien blijken principes van het ene deelgebied vaak verrassend goed van toepassing te zijn op een ander gebied. Het onderzoek naar dergelijke principes staat nog in de kinderschoenen, maar biedt belangrijke perspektieven.

Referenties:

[Akin81] Akin, O. and Rao, D. R., Efficient computer-user interface in electronic mail systems, report no. CMU-CS-81-140, Carnegie-Mellon University, Pittsburg, 1981.

[Allen81] Allen, T., Nix, R., and Perlis, A., PEN: a Hierarchical Document Editor, in [SIGPL81].

[Burk80] Burkhart, H. and Nievergelt, J., Structure-Oriented Editors, Berichte Inst. Inf. ETH Zuerich nr. 38.

[Burk81] Burkhart, H., Konzepte zur Systematisierung der Benutzerschnitstelle in Interaktiven Systemen und Ihre Anwendung auf den Entwurf von Editoren, Berichte Inst. Inf. ETH Zuerich, nr. 43.

[Benn72] Bennet, J.L., The User Interface in Interactive Systems, Annual Review of Information Systems, 7 (1972).

[BYTE81] BYTE Smalltalk issue, BYTE 6,8 (Aug 1981).

[CACM81] ACM Forum, On Natural Language and Computer Systems, CACM 24,6 (June 1981).

[Card80] Card, S.K., Moran, Th.P., and Newell, A., The Keystroke-Level Model for User Performance Time with Interactive Systems, CACM 23,7 (July 1980).

[Carg81] Cargill, T.A., Full-Screen Editing in a Hostile Environment, Software, Practice and Experience, 11,9 (Sept 1981), pp. 975-981.

[Cham81] Chamberlin, D.D., King, J.C., Slutz, D.R., Todd, S.J.P., and Wade, B.W., JANUS: An Interactive System for Document Composition, in [SIGPL81].

[Charl81] Charlton, C.C. and Lang, P.H., Editors: Two for the Price of One, Software, Practice and Experience, 11,2 (Feb 1981), pp. 195-202.

[Codd74] Codd, E.F., Seven Steps to Rendezvous with the User, in [Klim74].

[Codd78] Codd, E.F., How about Recently, in [Shnei78], pp. 3-28.

[Coul76] Coulouris, G.F., Durham, I., Hutchinson, J.R., Patel, M.H., Reeves, T., and Winderbank, D.G., The Design and Implementation of an Interactive Document Editor, Software, Practice and Experience, 6,2 (Apr-June 1976), pp. 271-279.

[Date81] Date, C.J., An Introduction to Database Systems, 3rd ed., Addison-Wesley (1977).

[Elli82] Elliot, B., Design of a Simple Screen Editor, Software, Practice and Experience 12,4 (April 1982).

[Embl81] Embley, D.W. and Nagy, G., Behavioral Aspects of Text Editors, Computing Surveys, 13,1 (March 1981), pp. 33-70.

[Fras80] Fraser, C.W., A Generalized Text Editor, CACM, 23,3 (March 1980).

[Fras81a] Fraser, C.W. and Lopez, A.A., Editing Data Structures, ACM TOPLAS, 3,2 (April 1981), pp. 115-125.

[Fras81b] Fraser, C.W., Syntax-directed editing of general data structures, In [SIGPL81], pp. 17-21.

[Furut82] Furuta, R., Scofield, J., Shaw, A., Document formatting systems: Survey, concepts and issues, Computing Surveys, 14,3 (Sept. 1982), pp. 417-472.

[Green78] Greenblatt, D. and Waxman, J., A Study of Three Database Query Languages, In [Shnei78], pp. 77-97.

[Herot80] Herot, C.F., Spatial Management of Data, ACM TODS, 5,4 (1980), pp. 493-514.

[Hamm81] Hammer, M., Ilson, R., Anderson, T., Good, M., Rosenstein, L., Niamir, B., Schoichet, S., and Gilbert, E., The implementation of Etude, an integrated document preparation system, in [SIGPL81].

[Iron72] Irons, E.T. and Djorup, F.M., A CRT Editing System, CACM 15,1 (Jan 1972), pp. 154-158.

[Kais80] Kaisler, St., The Agency Personal Information System, in [SIGSM80], pp. 114-125.

[Kay77] Kay, A. and Goldberg, A., Personal Dynamic Media, IEEE Computer, March 1977.

[Klim74] Klimbie, J.W., and Koffeman, K.I., (eds), Data Base Management, Proc. IFIP TC-2 Working Conf. on Data Base Management Systems, Cargese, Corsica, North Holland, 1974.

[Ledg80] Ledgard, H., Whiteside, J.A., Singer, A., and Seymour, W., The Natural Language of Interactive Systems. CACM 23,10 (Oct. 1980), pp. 556-563.

[Lefk79] Lefkovits, H.C., A status report on the activities of the Codasyl End User Facilities Committee, ACM SIGMOD Record, 10,2 & 3 (Aug. 1979).

[Mart73] Martin, Th.A., The User Interface in Interactive Systems, Annual Review of Information Systems, 8 (1973).

[McLeod77] McLeod, I.A., Design and Implementation of a Display Oriented Text Editor, Software, Practice and Experience, 7,6 (Nov-Dec 1977), pp. 771-778.

[Meyr82] Meyrowitz, N., and Van Dam, A., Interactive diting systems, part I and II, Computing Surveys, 14,3 (Sept. 1982), pp. 321-415.

[Palm79] Palme, J., A Human-Computer Interface for Non-Computer Specialists, Software, Practice and Experience, 9,9 (Sept 1979), pp. 741-747.

[Rams80] Ramsdell, R.E., The Power of VisiCalc, Byte, 5,11 (Nov 1980).

[Reid81] Reid, B.K., and Hanson, D., An Annotated Bibliography of Background Material on Text Manipulation, in [SIGPL81].

[Reis77] Reisner, Ph., Use of Psychological Experimentation as an Aid to Development of a Query Language, IEEE Trans. on Software Eng., 3,3 (May 1977), pp. 218-229.

[Reis81] Reisner, Ph., Human Factor Studies of Database Query Languages, Computing Surveys, 13,1 (March 1981).

[Rober79] Robertson, G., McCracken, D. and Newll, A., The ZOG approach to man-machine communication, Report no. CMU-CS-79-148, Carnegie-Mellon University, Pittsburg, Oct. 1979.

[Russ79] Russel, P., The Brain Book, Routledge and Kegan Paul, London, 1979.

[Shnei78] Shneiderman, B. (Ed), Databases: Improving Usability and Responsiveness, Academic Press, New York, 1978.

[Shnei80] Shneiderman, B., Software Psychology, Human Factors in Computer and Information Systems, Winthrop Publishers, Cambridge, Mass., 1980.

[SIGPL81] Proceedings of the ACM SIGPLAN/SIGOA Symposium on Text Manipulation, SIGPLAN Notices, 16,6 (June 1981).

[SIGSM80] Proc. Third Symp. on Small Systems, Sigsmall Newsletter 6,2 (Sept 1980).

[Snee78] Sneeringer, J., User-interface Design for Text Editing: A Case Study, Software, Practice and Experience, 8,5 (Sept-Oct 1978), pp. 543-557.

[Snod80] Snoddgrass, R., A sophisticated microcomputer user interface, in [SIGSM80], pp. 97-107.

[Stall81] Stallman, R.M., EMACS, the extensible, customizable, self-documenting display editor, in [SIGPL81].

[Strom81] Strömfors, O. and Jonesjö, L., The implementation and experiences of a structure-oriented editor, in [SIGPL81].

[Tsich82] Tsichritzis, D., Form management, CACM, 25,7 (July 1982), pp. 453-478.

[VDam71] Van Dam, A. and Rice, D.E., On-Line Text-Editing: A Survey, ACM Computing Surveys, 3,3 (Sept. 1971), pp. 93-114.

[Zloof77a] Zloof, M.M., Query by Example: A Data Base Language, IBM Syst. J., 16,4 (1977).

[Zloof77b] Zloof, M.M. and de Jong, S.P., The System for Business Automation (SBA): Programming Language, CACM, 20,6 (June 1977).

[Zloof78] Zloof, M.M., Design Aspects of the Query-by-Example Data Base Management Language, in [Shnei78].

[Zloof81] Zloof, M.M., QBE/OBE: A Language for office and business automation, IEEE Computer, 14,5 (May 1981), pp. 13-22.