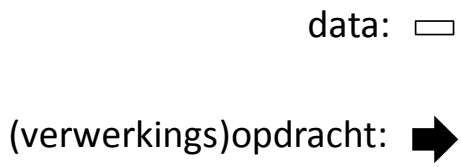
Omhakken!

Pieter Wisse

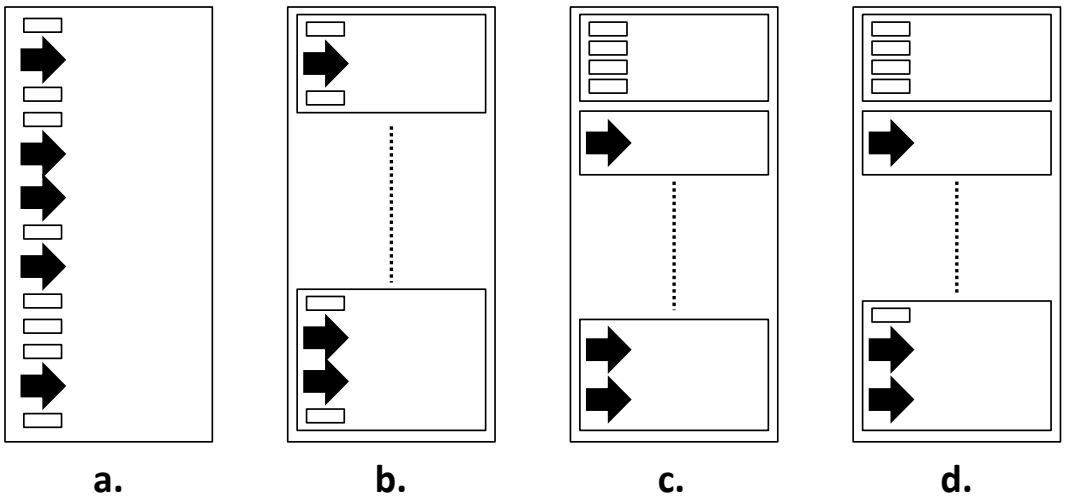
Ik heb al eens beschreven, hoe gevaar van een – overdrachtelijk gesproken – informatiemoeras dreigt. Sterker nog, her en der gaat het vooralsnog verder mis; zie Uit het moeras van stam- en basisgegevens (2011). De gemankeerde opvatting over wat in het Engels master data heet, illustreer ik hier in zo algemeen mogelijke termen met een beknopte geschiedenis van – streven naar – doelmatigheid in programmatuur(ontwikkeling). Stellig, zeg ook maar simplificerend, doe ik op sommige punten nauwkeurigheid enig geweld aan, maar de lijn klopt volgens mij. Uiteraard vermeld ik tevens – bijdrage aan – de oplossing.

Een computer werkt volgens onderscheid tussen object en proces. In het geval van een computer heet een object ook wel data(element). Een proces betreft zgn (data)verwerking volgens een opgegeven opdracht (lees ook: instructie). Van een verwerkingsgang is data invoer (grondstof) èn uitvoer (resultaat). In figuur 1 stel ik symbolen voor data (enkelvoud) en (verwerkings)opdracht voor.



figuur 1: gebruikte symbolen.

Voor het vroege stadium van computergebruik laat zich de verdeling van data (meervoud) en opdrachten in een willekeurig (computer)programma als volgt karikaturaal schetsen. Benodigde data staat steeds beschikbaar onmiddellijk vóór de opdracht voor de verwerking ervan tot ... data. Ik ga verderop door op – het belang van – context. Voor een programma met een opzet zoals figuur 2.a geldt in elk geval dat een bepaalde opdracht de relevante context – impliciet of niet, dat is nu eenmaal zo – vormt voor de data die specifiek ervoor is/zijn opgegeven.



figuur 2: ontwikkeling van modulariteit van programma.

Naarmate een programma uitgebreider is, kan indien mogelijk een onderverdeling overzichtelijkheid, beheersbaarheid enz. bevorderen: modulariteit door subprogramma's (lees ook: subroutines). Dat beginsel is wederom netzo karikaturaal weergegeven in figuur 2.b.

Latere moeilijkheden met zgn semantische interoperabiliteit zijn m.i. grotendeels herleidbaar tot de volgende, nota bene ook nog zeer vroege, stap in structuurontwikkeling van computerprogrammatuur. Een programma begint dan met een aparte module voor de data. Bijgevolg bevatten overige modulen louter opdrachten, zoals figuur 2.c wederom sterk gestileerd toont.

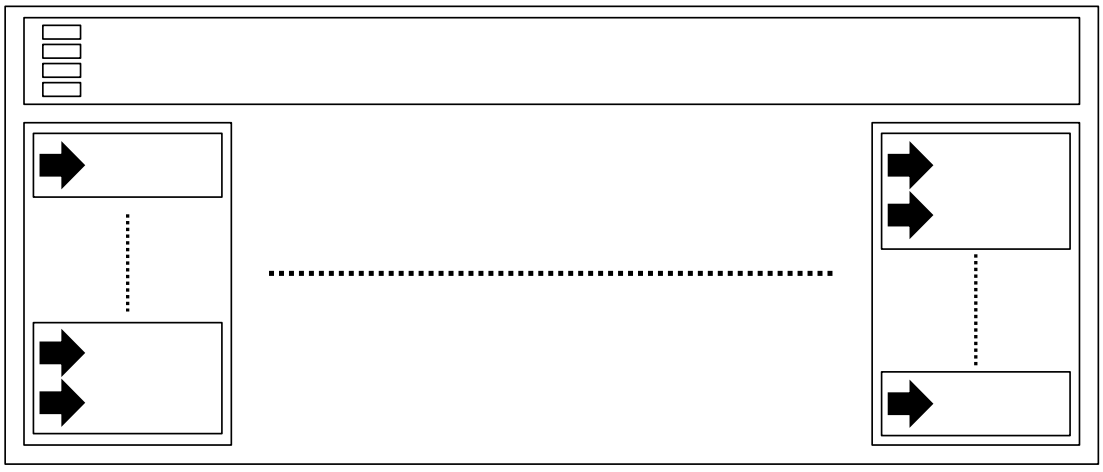
De maatregel om aparte modulen voor data resp. opdrachten in te richten lijkt evident voordelig. Zo behoeft dezèlfde data niet opnieuw voor verschillende opdrachten opgegeven te worden.

Het is echter de vraag, of data (meervoud) met dezelfde aanduiding ook inderdaad voor verschillende opdrachten steeds dezelfde data ... is.

Dat is waarschijnlijk zolang zoiets als het werkingsdomein van het programma in kwestie beperkt blijft. Daarbinnen houdt bepaalde data dan een vaste betekenis. Dat is met vroege toepassingen van computers als het ware vanzelfsprekend aan de orde. Daarbij gaat het doorgaans om berekeningen. En getallen vertegenwoordigen vergaande abstracties.

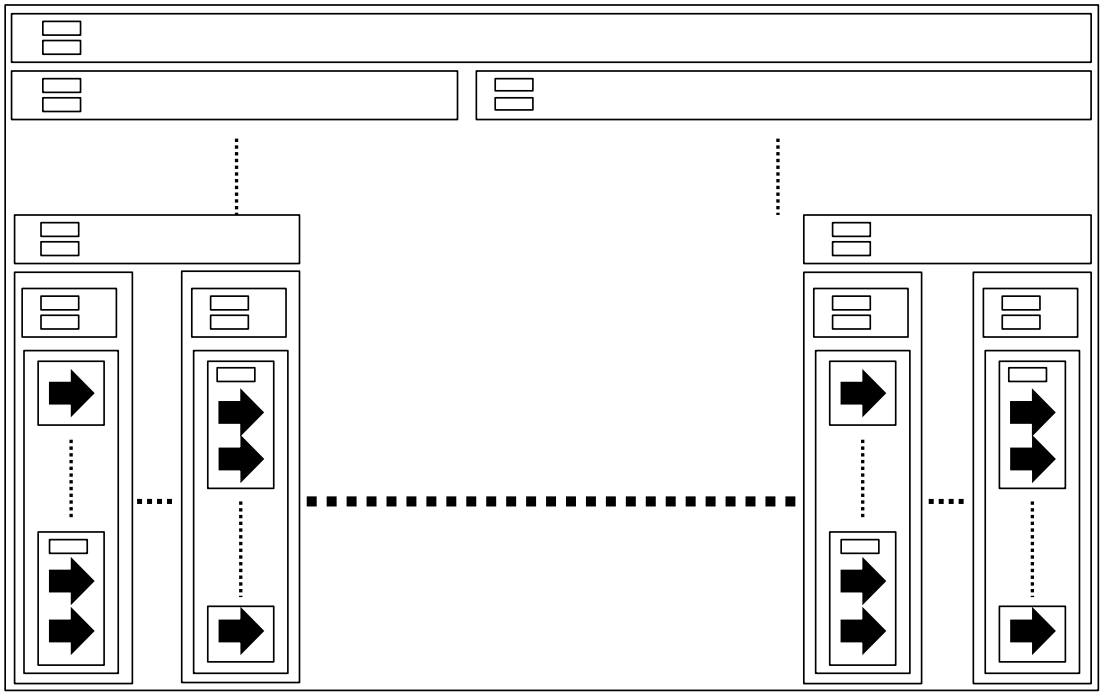
Er bestond destijds overigens wel degelijk besef, dat zulke gelijkheid niet absoluut geldig is. Daarom was er de mogelijkheid om in een opdracht(en)module toch tevens data op te nemen; het bereik van geldigheid ervan blijft beperkt tot de module in kwestie (en vervangt daarvoor wat met dezelfde aanduiding voor die data in de aparte datamodule staat opgegeven; zie figuur 2.d).

Hoe dan ook, op de schaal van een ènkel programma geldt/-en data in de aparte datamodule reeds als zgn stam-/basisgegevens. De volgende stap naar gemeenschappelijkheid van data onttrekt de opgave, het beheer enz. idealiter volledig aan een specifiek programma. Er komt een apart 'honk' voor data: database. Daarmee moeten allerlei programma's het doen, zie figuur 3.



figuur 3: geïdealiseerde database.

Met verschillende programma's groeit de kans, dat betekenissen variëren. De noodzaak van differentiatie leidt dan tot een hiërarchie van data(bases) zoals figuur 2.d aangeeft voor een ènkel programma. Let wel, dat stam- resp. basisgegevens betrekkelijk zijn. Met figuur 4 heb ik geprobeerd zo'n structuur op hoofdlijn weer te geven.



figuur 4: voortgezette veralgemenisering, met in omgekeerde richting ruimte voor afwijkingen.

Vooralsnog blijft het doel onder de noemer van stam-basisgegevens een opzet zoals figuur 3 laat zien. Er bestaat nog nauwelijks besef van wisselwerking. Naarmate de spreiding van – doelen met – programma's toeneemt, geldt echter voor minder data (meervoud) steeds dezelfde betekenis. Daar komt bij, dat computers verhoudingsgewijs ook minder gebruikt worden voor berekeningen – met getallen – en alsmaar méér voor opslag en communicatie, dwz voor tekens die niet vanwege hun abstracte aard praktisch algemeen geldig zijn, maar waarvan juist concreetheid geborgd moet zijn en blijven.

Praktisch houdt verruiming van schaal een kwalitatieve verandering in. Vooruit, voor steeds een ènkel (reken)programma gaat de opzet volgens figuur 2.c vrijwel altijd op. Maar dat laat zich niet òpschalen volgens figuur 3. En de opzet à la figuur 4 biedt evenmin oplossing, want zo'n hiërarchie veroorlooft slechts een enkele stam- annex basislijn. Allerlei reële betekenissen vallen daarbuiten, zodat ze alsnog op het laagste niveau opgegeven moeten zijn met vermenigvuldiging en juist navenante àfbreuk van doelmatigheid van dien.

Structurele oplossing vergt allereerst problematisering van de – alom impliciet geraakte – veronderstelling, dat data en proces onderling ònafhankelijk zijn. Wel beschouwd blijkt daarop de klassieke natuurkunde gebaseerd. De spreekwoordelijke oorzaak is de bewegende biljartbal. Wat heeft de botsing met een stilliggende biljartbal voor gevolg?

Semiotiek (Nederlands: tekenleer) is kwalitatief echter ànders. Een teken lijkt dus niet op een biljartbal in de zin dat er iets op een waarnemer botst. Daarentegen bepaalt de waarnemer mede wat z/hij op enig moment als teken laat tellen. Dat(a) hangt van het waarnemingsproces àf, meer in het bijzonder van het motief (lees ook: interesse) van een bepaalde waarnemer op een bepaald tijdstip en zodoende in wat z/hij ervaart als bepaalde situatie. Daarom kan invoer niet van verwerker gescheiden opgevat zijn, laat staan strikt. En wat ogenschijnlijk dezelfde invoer is, kan voor verschillende waarnemers – of dezelfde waarnemer in verschillende situaties – steeds een ànder teken zijn, met hun resp. haar/zijn dienovereenkomstig variërend

gedrag als uitvoer. Het lineaire schema van invoer-productie-uitvoer klopt domweg niet voor gedrag volgens semiosis.

Ik ga hier verder geen uitleg van subjectief situationisme resp. Metapatroon geven. Zie elders voor – zeer uitgebreide – documentatie, o.a. Metapatroon: handboek stelselmatig informatieverkeer. Ik herhaal hier slechts dat Metapatroon consequent breekt met absoluut onderscheid tussen data en proces. Aan daadwerkelijke data c.q. proces ligt altijd relativiteit ten grondslag volgens samenloop van ... data en proces (met volgens Metapatroon als kunstgreep voor grens een zgn horizon). En ter verbijzondering dragen data en proces omgekeerd bij aan eventuele vèrdere samenloop, enzovoort. Nota bene, op die manier bestaat er géén aparte categorie data onder de noemer van stam-/basisgegevens. Indien nodig en voldoende zgn contextueel verbijzonderd op de relevante stelselschaal is èlke data resp. proces enkelvoudig (lees ook: uniek e.d.) bepaald.

Iedereen die er ook maar even bij stil staat, erkent stellig voor eigen gedrag de betrekkelijkheid ermee van tekens (lees ook: informatie). Voor informatiekunde – ja, daar staat toch ècht informatie – schiet het met de noodzakelijke paradigmawissel echter helaas nog niet op. De averechtse vergelijking met stam enz. is eerder dan ook diep verworteld geraakt in klassiek-natuurkundige kennisleer. Maar dat is voor informatieverkeer op maatschappelijke schaal véél en véél te simpel.