Notities bij het vak Informatiesystemen

Maarten Fokkinga

November 1994

Deze bundel bevat een aantal notities die ik in de loop van het trimester gemaakt heb naar aanleiding van mijn overpeinzingen bij het vak Informatiesystemen.

Over de constructie van Dataflowdiagrammen oftewel: GHVs voor DFDs

Maarten Fokkinga

Versie van November 1994

Inleiding Bij het vak Informatiesystemen is een van de leerdoelen het kunnen construeren van Dataflow diagrammen (DFD). In deze notitie doe ik een poging hiervoor een *gewenst handelingsverloop* (GHV) op te stellen. Dit doe ik door de Toetsopgave 'Maak een DFD voor de casus Schaaktoernooi' (oktober 1994) stap voor stap uit te werken.

* * *

Met opzet probeer ik aan te sluiten bij de standaarduitwerking die bij de Toets is gemaakt. Alleen omwille van de duidelijkheid zal ik zo hier en daar van de naamgeving of structurering afwijken.

Entiteiten Laten we het te ontwikkelen informatiesysteem A noemen (van: toernooi-Administratie). Allereerst bepalen we de externe entiteiten, dat zijn de entiteiten die met A een uitwisseling van gegevens hebben. In de casus worden de volgende onderscheiden:

S (= speler)
T (= toernooileiding = toernooi-organisatie)
W (= wedstrijdleiding)

Het is mijns inziens een kwestie van smaak of je T en W identificeert of onderscheidt, en of je eventueel nog een andere externe entiteit P (= prikbord) onderscheidt. We zullen verderop zien dat er nog een externe entiteit nodig is, namelijk K (= klok).

Contextdiagram De volgende stap is het construeren van een contextdiagram (CD). Dit doe ik volgens een eerder beschreven manier [1]. Het resultaat daarvan is het volgende lijstje van interacties tussen het administratiesyteem en de externe entiteiten. De beschrijving 'in: ... ' definieert de data naar A toe; de beschrijving 'uit: ... ' definieert de data uit A komend. (Bij sommige interacties is 'in:' of 'uit:' overbodig, omdat er slechts één datastroom is.) Na het lijstje volgt nog een serie opmerkingen er over.

0. $S \rightarrow A$ *inschrijving van een speler*
in: spelergegevens
1. $A \rightarrow S$ *uitreiking inschrijfnummer en het speelschema*
uit: inschrijfnummer en het speelschema
2. $T \rightarrow A$ *invoering van het speelschema*
in: speelschema
3. $T \rightarrow A$ *invoering van de uitslag van een partij*
in: uitslag
4. $K \rightarrow A \rightarrow T$ *ronde-voorbereiding*
in: trigger 'uur voor ronde',
uit: paringsgegevens en uitslagbriefjes
5. $W \rightarrow A \rightarrow T$ *bepaling volgende toernooistand*
in: trigger 'alle uitslagen binnen',
uit: toernooistand en uitslagenoverzicht
6. $W \rightarrow A \rightarrow T$ *beeindiging van het toernooi*
in: trigger 'bepaal prijswinnaars',
uit: individuele en teamwinnaars

Interacties 0 en 1 hadden best gecombineerd kunnen worden tot één interactie:

0,1. $S \rightarrow A \rightarrow$S *inschrijving van een speler*
in: spelergegevens
uit: inschrijfnummer en het speelschema

Zonder deze combinatie blijkt er straks een trigger nodig te zijn; omwille van de leerzaamheid heb ik die keuze gemaakt.

Het is mijns inziens discutabel of het speelschema in A ingevoerd moet worden (interactie 2) en gereproduceerd moet worden (interactie 1). Aangezien het speelschema 'altijd door de toernooileiding wordt vastgesteld' is een copieermachine een handiger reproductiemechanisme dan een administratiesysteem.

De klok K is in punt 4 als externe entiteit gepostuleerd omdat, per afspraak, informatiesystemen voor alle output een trigger moet hebben. Een alternatief voor K is W of T.

Door een bolletje te tekenen om ieder van de letters A, K, S, T, W en bolletjes met gelijke letters erin verschillend te nummeren ("nummer *zoveel* van in totaal *zoveel*"), onstaat er een echt CD. We zullen verderop slechts letter A met een subscript nummeren: $A_0, A_1 \ldots$. Overigens, het is bij contextdiagrammen gebruikelijk om voor het informatiesysteem één bolletje te tekenen, met éénmaal de letter A erin. In bovenstaand lijstje krijg je dan een heel langgerekt bolletje dat alle zeven voorkomens van A bevat.

Procesen We gaan nu het ene proces A verder uit-detailleren in een aantal sub-processen, en eventueel interne stores toevoegen. De meest belangrijke opmerking van deze notitie is:

> *precies* de verschillende voorkomens van A in het CD hierboven zijn, of worden, de processen in het DFD.

We noemen de verschillende voorkomens van A ($A_0, A_1, \ldots$) vanaf nu 'processen'. Wat we verder doen is: bovenstaande opsomming van interacties tussen de A_i en de buitenwereld uitbreiden (met interacties tussen de A_i's onderling en met te introduceren stores) en wel op zodanige manier dat:

> ieder proces voldoende triggers krijgt om, zonder verder tijdsbesef, op het juiste moment de output te produceren, en
>
> ieder proces voldoende extra input van stores krijgt om, zonder verder geheugen, de juiste output te produceren.

Het valt te verwachten dat een interactie zonder output, zoals interactie 0, nu zijn input opslaat in een store; zou dat niet gebeuren, dan is de input tamelijk zinloos geweest (tenzij die input louter als trigger voor volgende interacties dient).

We markeren de uitbreidingen met •. Stores worden on-the-fly gepostuleerd. De uitbreiding van bovenstaand lijstje ziet er dan als volgt uit:

0. $S \rightarrow A_0$ *inschrijving van een speler*
   in: spelergegevens
   •uit: spelergegevens, naar: store *speler*
1. $A_1 \rightarrow S$ *uitreiking inschrijfnummer en het speelschema*
   uit: inschrijfnummer en het speelschema
   •in: nieuw inschrijfnummer, vanuit: store *speler*
   •in: schema, vanuit: store *schema*
   •in: trigger, vanuit: A_0
2. $T \rightarrow A_2$ *invoering van het speelschema*
   in: speelschema
   •uit: schema, naar: store *schema*
3. $T \rightarrow A_3$ *invoering van de uitslag van een partij*
   in: uitslag
   •uit: uitslag, naar: store *uitslag*
4. $\mathrm{K} \rightarrow A_4 \rightarrow T$ *ronde-voorbereiding*
   in: trigger 'uur voor ronde',
   uit: paringsgegevens en uitslagbriefjes
   •in: gegevens vanuit: store *paring*, *uitslag*
5. $W \rightarrow A_5 \rightarrow T$ *bepaling volgende toernooistand*
   in: trigger 'alle uitslagen binnen',
   uit: toernooistand en uitslagoverzicht
   •in: gegevens vanuit: store *uitslag*
6. $W \rightarrow A_6 \rightarrow T$ *beeindiging van het toernooi*
   in: trigger 'bepaal prijswinnaars',
   uit: individuele en teamwinnaars
   •in: gegevens vanuit: store *uitslag*, *speler*

Er zijn diverse keuzen gemaakt. We hadden ook naast store *uitslag* nog een extra store *stand* kunnen invoeren, die in interacties 4 en 6 gebruikt wordt in plaats van *uitslag* , en in interactie 5 bijgewerkt wordt. Net zo hadden we naast store *speler* een extra store *nummer* kunnen invoeren, die in interactie 1 gebruikt wordt in plaats van *speler* (en in interactie 2 geïnitialiseerd wordt).

Het gevraagde DFD onstaat nu door weer bolletjes te tekenen om de letters, de extra in's en uit's met pijlen aan te geven, en ook voor iedere store een plaatje te tekenen. In plaats van de weinig suggestieve namen ' A_0 ', ' A_1 ', ... kunnen we ook de beschrijvingen nemen die we bij de interacties hebben gezet.

Besluit Naar mijn idee is in het studiemateriaal van Informatiesystemen nergens vermeld dat de processen in een DFD eenduidig corresponderen met de interacties die uit de casus genoemd (of begrepen kunnen) worden.

Omdat het doorgeven van een trigger "kortgesloten" kan worden, heeft ieder DFD dat volgens het GHV geconstrueerd is een bijzondere structuur: zoiets als $A_i \to A_j \to A_k$ komt niet voor.

De event-lists uit het vak Informatiesystemen lijken nogal op de 'interactie-opsommingen' die ik gegeven heb. Ruwweg gezegd is dit de overeenkomst: in de kolom 'gebeurtenis' van een event-list staat de informele beschrijving van een interactie uit de interactie-opsomming, in de kolom 'actie' van een event-list staat de aanduiding A_i , en de 'indirecte output' in een event list is ongeveer het gegeven dat bij de " •uit: " in een store opgeslagen wordt. (Alleen met de kolom 'statusovergang' weet ik geen raad; en ik betwijfel de wel-gedefinieerdheid van dat begrip.) Mijn conclusie is dat *zoiets als* een event-list méér een noodzakelijk hulpmiddel is om een DFD te construeren, dan dat het een middel is voor de afstemming van verschillende modellen (CD, DFD, ER, Rel, SD).

Referenties

[1] M.M. Fokkinga. Constructie van contextdiagrammen. September 1994.

Algoritmische constructie van ER-modellen

Maarten Fokkinga

Versie van November 1994
SCHETS — MISSCHIEN NOG NIET FOUTLOOS

Inleiding Bij het vak Informatiesystemen komen onder andere ER-modellen aan bod, maar zonder enige formele semantiek. Deze verhandeling geeft een semantiek die erg dicht tegen de syntaxis aan ligt, maar algemeen bruikbaar lijkt. Ruw gezegd representeert een ER-model, volgens deze semantiek, een functionele afhankelijksheidsrelatie op een compacte manier. Omgekeerd geldt, als stelling, dat er voor iedere functionele afhankelijksrelatie zo een corresponderend ER-model bestaat.

Bij Toets I van het vak Informatiesytemen (oktober 1994) wordt een ER-model gevraagd voor de casus *Van de Weg AF*. Het ER-model uit de standaard uitwerking wordt door het algoritme van de stelling opgeleverd; als input zijn daarvoor nodig: de verzameling *ongeïnterpreteerde* attributen en hun *ongeïnterpreteerde* functionele afhankelijkheidsrelatie — deze beiden moeten door interpretatie uit de casus gehaald worden. Ik vermoed dat veel ER-modellen op deze algoritmische wijze geconstrueerd kunnen worden: de modelbouwer hoeft de gegevens van de casus alleen maar te interpreteren om de verzameling van attributen te bepalen en hun functionele afhankelijkheden; verder niet.

Met deze studie hoop ik meer begrip te krijgen van het vak. Ik houd me aanbevolen voor commentaar.

Om het een en ander te bewijzen, ben ik gedwongen eerst de begrippen formeel te definiëren. Als het goed is, vind je hier geen grote verrassingen.

Allereerst enige terminologie en notatie. Vaak zullen we een verzameling $\mathcal{A}$ gegeven veronderstellen. De elementen ervan noemen we dan **attributen** en we laten $a, b, \ldots$ daarover varieren en $A, B, \ldots$ over deelverzamelingen: $a \in A \subseteq \mathcal{A}$.

Definities (FA-relaties) Laat $\mathcal{A}$ een verzameling zijn, en $\multimap\!\!\bullet\!\!\rightarrow$ een binaire (infix geschreven) relatie op deelverzamelingen van $\mathcal{A}$. Dan heet $\multimap\!\!\bullet\!\!\rightarrow$ een **FA-relatie** als die reflexief en transitief is en de volgende *monotonie-eigenschap* heeft:

$$A \supseteq A' \multimap\!\!\bullet\!\!\rightarrow B' \supseteq B \quad \Rightarrow \quad A \multimap\!\!\bullet\!\!\rightarrow B \quad .$$

Functionele afhankelijkheidsrelaties zijn FA-relaties; vandaar de naamgeving 'FA'.
Een $\multimap\!\!\bullet\!\!\rightarrow$**-sleutel** van A is een minimale verzameling B met $B \multimap\!\!\bullet\!\!\rightarrow A$. Een $\multimap\!\!\bullet\!\!\rightarrow$**-basis** is een relatie $\multimap\!\!\bullet\!\!\rightarrow'$ waarvan de kleinst omvattende FA-relatie precies $\multimap\!\!\bullet\!\!\rightarrow$ is.

Definitie (ER-model) Laat $\mathcal{A}$ een attribuutverzameling zijn. Een **ER-model** $\mathcal{M}$ over $\mathcal{A}$ is, in deze verhandeling, een gerichte graaf (met pijlen $\longrightarrow$ en $\longmapsto$) tesamen met twee functies at en sl die voldoen aan onderstaande eigenschappen (a), (b), (c).

Een knoop in $\mathcal{M}$ heet **entiteit**. We laten $E, E', \ldots$ varieren over entiteiten. We zeggen dat $at\,E$ de **attributen van** E zijn, en dat $sl\,E$ de attributen van E zijn die een sleutelmerk dragen. Een entiteit heet **leeg** wanneer $at\,E$ leeg is.

Een kant in $\mathcal{M}$ heet **relatie** en wordt genoteerd met $\longrightarrow$. Een relatie is optioneel gemerkt met het sleutelmerk, en wordt dan genoteerd met $\longmapsto$. Dus de geldigheid van $E \longmapsto E'$ impliceert ook $E \longrightarrow E'$. Zoals gebruikelijk is $\longrightarrow^*$ de reflexieve transitieve afsluiting van $\longrightarrow$.

Opdat $\mathcal{M}$ een ER-model is, moet ook nog het volgende gelden:

(a) Voor iedere entiteit E is $sl\,E \subseteq at\,E \subseteq \mathcal{A}$

(b) De attribuutverzamelingen $at\,E$ zijn onderling disjunct

(c) De $\longmapsto$-relaties vormen geen cycle .

Is hieraan voldaan dan zeggen we ook wel dat $\mathcal{M}$ syntactisch correct is.

Verband met echte ER-modellen Het verband met de echte ER-modellen (uit het vak Informatiesystemen) is als volgt. Een relatie $E \longrightarrow E'$ hier betekent E $\underline{\text{(1,1)}\ \ r\ \ \text{(0,}n\text{)}}$ E' (voor een of andere r) in I.S., en omgekeerd. Een relatie E $\underline{\text{(1,1)}\ \ r\ \ \text{(1,1)}}$ E' wordt in ons begrip van ER-model weergegeven door $E \longrightarrow E'$ tesamen met $E' \longrightarrow E$.

Relaties met cardinaliteiten $(\_, n)$—$(\_, n)$ bestaan niet in ons model; ze komen in het echt ook weinig voor. Evenmin kunnen we tussen twee entiteiten E en E' verschillende relaties $r, r', \ldots$ weergeven. Het lijkt niet moeilijk om de theorie te veralgemenen zo dat FA-relaties $\multimap\!\!\bullet\!\!\rightarrow$ en ER-relaties $\longrightarrow$ gelabeld zijn met symboolrijen gevormd uit een symboolverzameling $\{r, r', \ldots\}$. Deze veralgemening laten we eenvoudigheidshalve achterwege. Bovendien komen entiteiten met meerdere relaties ertussen niet veel voor in de casussen die ik heb gezien.

Definitie (Semantiek van een ER-model) Laat $\mathcal{M}$ een ER-model zijn over $\mathcal{A}$. Allereerst bepaalt $\mathcal{M}$ een functie Sl, die aan iedere entiteit een deelverzameling van $\mathcal{A}$ toewijst. Verzameling $Sl\,E$ heet de **sleutel** van E in $\mathcal{M}$; hiertoe behoren, ruw gezegd, de attributen van E die een sleutelmerk hebben tesamen met al dergelijke attributen die via sleutel-relaties uit E bereikbaar zijn:

$$Sl\,E \quad = \quad sl\,E \quad \cup \quad \bigcup \{ Sl\,E' \mid \text{er is een } E \longmapsto E' \text{ relatie in } \mathcal{M} \} \quad .$$

Omdat in een ER-model de relatie $\longmapsto$ geen cycle bevat, is deze definitie niet circulair.

Ten tweede bepaalt $\mathcal{M}$ een FA-relatie $\multimap\!\!\circ\!\!\rightarrow$ over $\mathcal{A}$. Relatie $\multimap\!\!\circ\!\!\rightarrow$ is, per definitie, de kleinste FA-relatie met:

$$Sl\,E \quad \multimap\!\!\circ\!\!\rightarrow \quad at\,E' \quad \text{voor elke } E, E' \text{ met } E \longrightarrow^* E' \text{ in } \mathcal{M} \quad .$$

Met andere woorden, relatie $\multimap\!\circ\!\!\rightarrow$ is als volgt inductief gedefinieerd:

$$
\begin{array}{lcl}
Sl\,E \multimap\!\circ\!\!\rightarrow at\,E & & \\
E \longrightarrow E' & \Rightarrow & Sl\,E \multimap\!\circ\!\!\rightarrow at\,E' \\
A \supseteq A' \multimap\!\circ\!\!\rightarrow B' \supseteq B & \Rightarrow & A \multimap\!\circ\!\!\rightarrow B \\
A \multimap\!\circ\!\!\rightarrow A & & \\
A \multimap\!\circ\!\!\rightarrow B \multimap\!\circ\!\!\rightarrow C & \Rightarrow & A \multimap\!\circ\!\!\rightarrow C \quad .
\end{array}
$$

Dus de eerste twee clausules samen definiëren een basis voor $\multimap\!\circ\!\!\rightarrow$. Bovendien volgt uit de eerste clausule dat $Sl\,E$ een $\multimap\!\circ\!\!\rightarrow$-sleutel van $at\,E$ omvat (en er zijn gevallen waarin $Sl\,E$ geen $\multimap\!\circ\!\!\rightarrow$-sleutel van E is: kies $\mathcal{M}$ zo dat $Sl\,E_0 = at\,E_0 \neq \emptyset$, terwijl $E_1 \longrightarrow E_0$ met $Sl\,E_1 = \emptyset$). We noemen $\multimap\!\circ\!\!\rightarrow$ de **semantiek** van $\mathcal{M}$, en we noemen twee ER-modellen **equivalent** wanneer ze dezelfde semantiek hebben. We noemen een ER-model **minimaal** wanneer er geen relatie en geen lege entiteit (met al z'n in- en uitgaande relaties) weggelaten kan worden zonder de semantiek te veranderen.

Definitie (Correctheid) Laat $\mathcal{A}$ gegeven zijn, alsmede een FA-relatie $\multimap\!\bullet\!\!\rightarrow$ en een ER-model $\mathcal{M}$ over $\mathcal{A}$. Dan heet $\mathcal{M}$ (semantisch) **correct** met betrekking tot $\multimap\!\bullet\!\!\rightarrow$ wanneer geldt:

(a) de vereniging van de $at\,E$'s is $\mathcal{A}$

(b) de semantiek $\multimap\!\circ\!\!\rightarrow$ van $\mathcal{M}$ is $\multimap\!\bullet\!\!\rightarrow$, en

(c) iedere entiteit is in 2de en 3de normaalvorm m.b.t. $\multimap\!\bullet\!\!\rightarrow$ (zie onder) .

Een entiteit E is *niet* in 2de normaalvorm als, ruw gezegd, een deel van E niet-triviaal afhangt van slechts een deel van $Sl\,E$; formeel:

$$\exists A, B :: \quad A \subseteq at\,E \;\wedge\; B \subset Sl\,E \;\wedge\; A \not\subseteq B \multimap\!\bullet\!\!\rightarrow A \quad .$$

Een entiteit E is *niet* in 3de normaalvorm als, ruw gezegd, een deel van E niet-triviaal afhangt van een deel buiten de sleutel dat zelf geen alternatieve sleutel is; formeel:

$$\exists A, B :: \quad A \subseteq at\,E \;\wedge\; B \subseteq (at\,E - Sl\,E) \;\wedge\; A \not\subseteq B \multimap\!\bullet\!\!\rightarrow A \;\wedge\; B \not\multimap\!\bullet\!\!\rightarrow Sl\,E \quad .$$

Als $\mathcal{M}$ correct is met betrekking tot $\multimap\!\bullet\!\!\rightarrow$, dan is $Sl\,E$ een $\multimap\!\bullet\!\!\rightarrow$-sleutel voor E.

Opmerking De voorwaarde 'dat zelf geen alternatieve sleutel is', bij de 3de normaalvorm, staat niet in het collegemateriaal. Er zijn wel voorbeelden van ER-modellen die, zonder deze toevoeging, fout zouden zijn. Bovendien lijkt mij deze voorwaarde in overeenstemming met het doel van de normaalvorm, namelijk het voorkomen van redundantie in de opslag van gegevens.

De correctheidseisen hier boven stemmen geheel overeen met wat ik in een eerder notitie ("Over de normering van ER-modellen", oktober 1994, naar aanleiding van Toets I bij het vak Informatiesystemen) correctheid van de entiteiten heb genoemd.

Stelling Er is een efficient algoritme dat, bij gegeven $\mathcal{A}$ en gegeven (basis voor een) FA-relatie $-\!\!\bullet\!\!\rightarrow$ over $\mathcal{A}$, een $-\!\!\bullet\!\!\rightarrow$-correct (en vermoedelijk: minimaal) ER-model $\mathcal{M}$ oplevert.

Opmerking Zonder de efficiëntie-voorwaarde is er een triviaal algoritme: som alle mogelijke ER-modellen over $\mathcal{A}$ op (dat zijn er eindig veel als $\mathcal{A}$ eindig is), en filter daaruit degene die correct zijn voor $-\!\!\bullet\!\!\rightarrow$. Dit algoritme vergt, lijkt mij, zoiets als 2^n veel stappen, waarbij n de grootte van $\mathcal{A}$ is.

De efficiënte constructie hieronder vergt, denk ik, kwadratisch veel stappen gemeten in het aantal elementen in de gegeven $-\!\!\bullet\!\!\rightarrow$-basis. In ieder geval, hoe kleiner de basis, hoe efficiënter de constructie. De constructie gaat in twee stappen: eerst de constructie van een correct model aan de hand van relatie $-\!\!\bullet\!\!\rightarrow$, en daarna een minimalisatie onafhankelijk van $-\!\!\bullet\!\!\rightarrow$.

Het minimalisatie-algoritme is nondeterministisch. Ik vermoed dat door steeds geschikte keuzen te maken elk $-\!\!\bullet\!\!\rightarrow$-correct minimaal ER-model en elk $-\!\!\bullet\!\!\rightarrow$-correct ER-model zonder lege entiteiten opgeleverd kan worden. Dit vermoeden laat ik verder onbewezen, in deze verhandeling.

Een correct model voor $-\!\!\bullet\!\!\rightarrow$ is, in feite, een compacte (en grafische!) weergave van $-\!\!\bullet\!\!\rightarrow$: in elk model dat door het algoritme wordt opgeleverd, is het aantal relaties $\longrightarrow$ hoogstens het aantal elementen $A -\!\!\bullet\!\!\rightarrow' B$ is in de gegeven basis voor $-\!\!\bullet\!\!\rightarrow$.

Bewijs van de Stelling Laat $-\!\!\bullet\!\!\rightarrow'$ een basis voor $-\!\!\bullet\!\!\rightarrow$ zijn. Neem voor ieder attribuut $a \in \mathcal{A}$ een entiteit E_a (met a als enig attribuut), en voor ieder element $A -\!\!\bullet\!\!\rightarrow' B$ in de basis een (lege!) entiteit $E_{A-\!\!\bullet\!\!\rightarrow' B}$, en definieer daarbij de functies at en sl en de relaties $\longrightarrow$ en $\longmapsto$ als volgt:

$$
\begin{array}{lcll}
at\, E_a & = & sl\, E_a & = \quad \{a\} \\
\text{(geen uitgaande relaties uit } E_a\text{)} & & & \\
at\, E_{A-\!\!\bullet\!\!\rightarrow' B} & = & sl\, E_{A-\!\!\bullet\!\!\rightarrow' B} & = \quad \emptyset \\
E_{A-\!\!\bullet\!\!\rightarrow' B} \longmapsto E_a & & \text{voor iedere } a \in A & \\
E_{A-\!\!\bullet\!\!\rightarrow' B} \longrightarrow E_b & & \text{voor iedere } b \in B & .
\end{array}
$$

Omdat de entiteiten zo eenvoudig zijn en er geen ketens van $\longrightarrow$ en $\longmapsto$ zijn, is het volgende erg gemakkelijk te verifiëren. Ten eerste, de constructie levert een syntactisch correct ER-model: (a) voor alle E geldt $sl\, E \subseteq at\, E \subseteq \mathcal{A}$, (b) de $at\, E$'s zijn onderling disjunct, en (c) de relaties $\longmapsto$ vormen geen cycle. Ten tweede, het ER-model is semantisch correct voor $-\!\!\bullet\!\!\rightarrow$: (a) de $at\, E$'s tesamen omvatten geheel $\mathcal{A}$, (b) de relaties $-\!\!\bullet\!\!\rightarrow$ en $-\!\!\circ\!\!\rightarrow$ vallen samen want voor iedere A, B met $A -\!\!\bullet\!\!\rightarrow' B$ geldt ook $A = Sl\, E_{A-\!\!\bullet\!\!\rightarrow' B} -\!\!\circ\!\!\rightarrow B$ (dus $-\!\!\circ\!\!\rightarrow$ omvat de basis $-\!\!\bullet\!\!\rightarrow'$), en (c) iedere E is in 2de en 3de normaalvorm.

Vervolgens transformeren we het model met twee transformatieregels die ieder de (syntactische en semantische) correctheid behouden, maar de grootte van het model verminderen. Regel 1 vermindert het aantal entiteiten, terwijl Regel 2 het aantal entiteiten niet wijzigt maar wel het aantal relaties in het model vermindert. Dus herhaalde toepasssing

verkleint het model steeds verder. Wanneer geen regel meer toepasbaar is, is er dus een $\multimap\!\bullet\!\rightarrow$-correct minimaal ER-model verkregen.

Nog te voltooien

Over de normering van ER-modellen

Maarten Fokkinga, oktober 1994

Bij Toets I (oktober 1994) van het vak Informatiesystemen luidt de beoordelingsnormering:

> Iedere correcte entiteit en relatie wordt met 3 punten gehonoreerd.
> Per entiteit aftrek als sleutel niet correct (1 punt), attributen niet correct (1 punt).
> Per relatie aftrek als cardinaliteit niet correct (1 punt).

Deze normering is weliswaar zeer eenvoudig toe te passen (en levert bij goedwillende interpretatie ook redelijke uitkomsten), maar is bij nadere inspectie nogal dubbelzinnig. Bijvoorbeeld, wordt een entiteit met een incorrect attribuut gehonoreerd met $3-1$ punt of met $0-1$ punt? En wat is eigenlijk een 'incorrect attribuut'? En levert een relatie met beide cardinaliteiten fout $3-2$ punten of $0-2$ of 0 punten (en worden in het laatste geval noch de relatie geteld, noch de fouten in de cardinaliteiten)? Tenslotte, je krijgt andere uitkomsten wanneer je uit gaat van de goede dingen (en die beloont), dan wanneer je uit gaat van de foute dingen (en die bestraft).

Om wat meer inzicht te krijgen in deze materie, en daarmee mijn eigen kennis te vergroten, presenteer ik onderstaande beschouwing. Zij geeft een 'theoretisch ideale' normering. Ik houd me aanbevolen voor commentaar.

Entiteiten Ik stel voor om bij de 'correctheid' van de entiteiten *niet* te letten op de naamgeving van de entiteiten, maar alleen op de attributen die ze bevatten: een entiteit is dus niets meer en niets minder dan een stel attributen. De entiteiten gezamelijk zijn dan correct wanneer alle volgende vragen positief beantwoord worden.

1. Komen alle attributen uit de casus ergens in een entiteit voor?
2. Komt geen attribuut twee- of meermalen voor?
3. Zijn, per entiteit, de attributen die als sleutel zijn aangegeven, inderdaad een sleutel voor die entiteit?
4. Is iedere entiteit, gezien als 'relatie' in het 'relationele model', in 3de normaalvorm? (In het bijzonder, zijn alle attributen van een entiteit "atomair", dus geen repeterende groep?)

Een vrij preciese maat voor de fout ten aanzien van punt 4 hierboven wordt gegeven door: het aantal extra entiteiten dat ontstaat tijdens normalisatie van de entiteit. Immers, iedere normalisatie-stap brengt nieuwe entiteiten voort. Punt 4 en 3 kunnen locaal, per entiteit, beoordeeld worden. Daarbij is het nodig dat de attributen een semantiek hebben (om functionele afhankelijkheden te bepalen); die semantiek wordt door hun naamgeving én de casus gesuggereerd.

Naast bovengenoemde fouten is er nog de fout van misleidende naamgeving. Op zich kan hier niets fout gaan, want entiteit-namen hebben geen andere semantiek dan een afkorting

te zijn voor een stel attributen. Maar misleidend is het wel, te meer daar de "foute" namen nogal eens namen zijn die ook —maar anders— voorkomen in de standaard uitwerking.

Een "teveel aan entiteiten" kan mijns inziens geen fout zijn bij de entiteiten. Zoiets manifesteert zich door (misschien) een fout volgens punt 3, en (waarschijnlijk) een fout bij de relaties.

Normering Voor de normering doe ik de volgende suggestie voor de relatieve zwaartes. (De getallen zijn maar voorbeelden.)

10 Voor alle foute naamgeving tesamen.
5 Voor elke fout tegen punt 1 en 2.
10 Voor elke fout tegen punt 3.
10 Voor elke entiteit te weinig bij een fout tegen punt 4.

Relaties Ik zie geen andere mogelijkheid dan de correctheid van een relatie

$$E_0 \ \underline{(a,b) \quad r \quad (c,d)} \ E_1$$

te definieren als: de cardinaliteiten (a, b) en (c, d) gelden voor de relatie r op E_0 en E_1. (Dus: voor iedere waarde e_0 zijn er tenminste a en ten hoogste b waarden e_1 zó dat $(e_0, e_1) \in r$, etcetera). Dit vereist dat de naam r een semantiek heeft; die wordt gesuggereerd door de naam r én de casus. Merk op dat er voor ieder tweetal entiteiten een correcte relatie is van cardinaliteit (0,N)—(0,N), genaamd *"heeft wel of niet iets te maken met"*. Dus, bepalend voor de correctheid van een relatie zijn:

de door de student gegeven naam van de relatie; en
de door de student gegeven entiteiten (= de attributen, niet de naamgeving).

Voorts bestaan er correcte relaties die in de standaarduitwerking niet genoteerd staan. Immers, bij de relaties:

$$E_1 \ \underline{(a,b) \quad R \quad (c,d)} \ E_2 \ \underline{(e,f) \quad S \quad (g,h)} \ E_3$$

is, per transitiviteit, ook de volgende relatie correct:

$$E_1 \ \underline{(a{\cdot}e, b{\cdot}f) \quad R{\circ}S \quad (c{\cdot}g, d{\cdot}h)} \ E_3 \quad .$$

Het is gebruikelijk (in dit vak verplicht???) om een minimaal stel relaties te noteren. Daaruit zijn dus via transitiviteit alle andere uit afleidbaar.

Normering Ik suggereer de volgende normering voor de relatieve zwaartes. Breid allereerst het student-model uit met zo min mogelijk relaties maar zo dat de uitbreiding wel volledig is, dat wil zeggen, dat alle gewenste relaties daaruit afleidbaar zijn. Dan:

10 Voor iedere zojuist toegevoegde relatie.
5 Voor iedere foute cardinaliteit (m,n); dus 10 voor een geheel foute relatie.
5 Voor iedere irrelevante relatie (onafhankelijk van de correctheid).
5 Voor iedere al afleidbare relatie (onafhankelijk van de correctheid).

Uitvoering Hier volgt een algoritme waarmee een totaalbeoordeling eenvoudig uitgerekend kan worden. Dit algoritme is niet specifiek voor dit vak, en in feite onafhankelijk van bovenstaande.

Het idee is de bovengegeven zwaartes (zoals 5 en 10) op te vatten als procentuele aftrek, *steeds van het nog resterende puntentotaal*, zoals bij rente-op-rente. Bijvoorbeeld, deze *cumulatieve aftrek* van achtereenvolgens 10%, 5%, 5%, 10%, 10% van initieel 100% levert:

$$100\% \times 90\% \times 95\% \times 95\% \times 90\% \times 90\% \quad = \quad 65.79225\% \quad .$$

De procentuele aftrek heeft als voordeel boven absolute aftrek dat herhaling van fouten steeds minder zwaar gaat tellen, en dat in het begin de fouten toch zwaar aangerekend kunnen worden. Bij cumulatieve aftrek van 10% krijg je, uitgaande van 10 punten initieel en afgerond tot hele getallen, de volgende eindcijfers:

# aftrek	0	1	2	3–4	5	6–7	8–9	10–13	14–18	19–28
eindcijfer	10	9	8	7	6	5	4	3	2	1

Met een programmeerbare calculator is iedere aftrek met één toetsaanslag uit te voeren. Je hoeft bij het corrigeren alleen maar iedere fout zó te merken dat de relatieve zwaarte ervan duidelijk is. De volgorde van aftrek doet er niet toe.

Over Statusdiagrammen en Dataflowdiagrammen

Maarten Fokkinga

Versie van 19 september 1994

Inleiding Bij het vak Informatiesystemen (de versie van september 1994) is een van de werkcollege-opdrachten om voor de casus Sportbond een statusdiagram te maken van een 'lid'. Hieronder geef ik enige overwegingen aan die daarbij een rol zouden moeten spelen (maar in het schriftelijke collegemateriaal niet genoemd worden). In het bijzonder geef ik een formeel verband aan tussen SD en DFD dat nagestreefd kan of zou moeten worden, en hoe events en condities formeel onderscheiden kunnen worden.

De standaard uitwerkingen van het DFD en SD voor de casus voldoen niet aan dat verband, maar het is wel mogelijk ze aan te passen zodat het verband er wel is. Dit leidt tot een alternatieve uitwerking (die ik, vanwege het formele verband, qualitatief beter vind).

Deze notitie is bedoeld als informeel discussiestuk voor verbetering van mijn eigen begrip (en wellicht ook verbetering van het vak en vakgebied). Ik doe heel wat beweringen waarvan een formeel bewijs vooralsnog ontbreekt, en die waarschijnlijk aangepast moeten voordat ze formeel bewezen kunnen worden. Ik ontvang graag suggesties voor verdere formalisatie.

De status van een lid Wat is de status van een lid? Een lid kan in de volgende toestanden zijn: thee drinkend, versufd, op vakantie zijnd, een wedstrijd spelend, onder de douche staand, enzovoorts. Het is overduidelijk dat in het kader van de casus deze toestanden niet alle even relevant zijn. Wat zijn dan eigenlijk wel de relevante toestanden? Eén antwoord is: die toestanden die *gezien de tekst van de casus* relevant zijn en onderscheiden worden. De relevantie is dan nog steeds een erg vaag begrip. Bijvoorbeeld: 'een wedstrijd spelend' lijkt relevant gezien de tekst van de casus, terwijl ik (en anderen met mij) dat toch niet als een van de toestanden van een lid wil zien.

Er is ook een ietwat formeler antwoord, wanneer we uitgaan van het bestaan van een DFD voor de casus. Namelijk: de gezamelijke gegevensopslagen in de DFD bepalen wat de 'leden' zijn, en we willen alleen toestanden van een lid onderscheiden als dat *op grond van de gegevensopslag in de DFD* mogelijk is. In de gegeven DFD (de standaard uitwerking) zijn er de volgende opslagen: spelergegevens, overtredingsgegevens, strafgegevens. Zoiets als 'thee drinkend' of 'een wedstrijd spelend' kan niet op grond van deze gegevensopslag onderscheiden worden. (Ik ga er vauit dat een DFD niet alleen een plaatje is maar ook semantiek heeft!!!)

Aldus kom ik tot de conclusie dat bij het gegeven DFD tenminste de volgende toestanden van een lid onderscheiden kunnen worden:

> competitie spelend
> een overtreding begaan hebbend maar nog niet bestraft zijnd
> bestraft met een (al of niet voorwaardelijke) schorsing.

De tweede toestand wordt niet in de standaard uitwerking genoemd. Eventueel kan de laatste toestand nog onderscheiden worden in twee toestanden: geschorst en voorwaardelijk geschorst. Dit is in de standaard uitwerking het geval.

De toestand 'gewoon lid' kan niet zonder meer onderscheiden worden omdat de DFD een model is voor de administratie van uitsluitend *competitie spelende* leden! Een formeel verband tussen DFD en SD kan natuurlijk alleen maar bestaan voor de doorsnee van wat ze modelleren. Of iets preciezer:

> *wanneer een SD de toestanden en toestandsovergangen vast legt van een stel stores van een DFD, dan bestaat er een formeel verband tussen SD en DFD.*

Events en acties Laten we nu aannemen dat uitsluitend de gezamelijke stores van de DFD de mogelijke toestanden van een lid bepalen (en daarnaast bepalen ze wellicht nog meer). Wat zijn dan de acties die een toestandsovergang bewerkstelligen? Mijn antwoord is: dat zijn dan de *processen* die de stores van inhoud kunnen veranderen. En de events zijn dan de *aankomsten van data* bij die processen. De condities zijn *eigenschappen van die data* op grond waarvan de processen variëren in de verandering die ze aanbrengen in de stores.

In de standaard DFD-uitwerking vinden we inderdaad vele malen het proces 'verwerken spelerslijst' als actie. De bijhorende event heet in de standaard SD-uitwerking 'inschrijving' terwijl ik die dus zou benoemen met 'aankomst spelerslijst' op grond van de dataflow-benoeming 'spelerslijst' in de DFD.

De transitie van 'competitie spelend lid' naar 'voorwaardelijk bestraft competitie spelend lid' is in de standaard SD-uitwerking:

> E: bestraffing
> C: 1 geel
> A: verwerken strafuitspraak

terwijl ik die, gezien de DFD, zou ik formuleren als:

> E: aankomst van: strafuitspraak
> C: de aangekomen data voldoet aan: het is een voorwaardelijke uitspraak
> A: verwerken strafuitspraak.

(In het echt zou ik de E en C benoeming afkorten tot datgene wat na de dubbele punt staat.) Met deze definitie van wat de acties, events en condities zijn, is het formeel mogelijk om events en condities te onderscheiden.

Gevolg Het nastreven van een formeel verband tussen SD en DFD kan leiden tot aanpassing van beide, en daarmee tot qualitatieve verbetering. (Dit is in het algemeen een nut van het beschrijven van eenzelfde verschijnsel met verschillende formalismen.) Ik zal hiervan een voorbeeld geven.

In de standaard SD-uitwerking is er een toestandsovergang:

van: voorwaardelijk bestraft competitie spelend lid,
naar: competitie spelend lid
E: verstrijken voorwaarde
C: 3 mnd verstreken
A: opheffen voorwaarde.

Het enige proces in de gegeven DFD waarmee ik deze actie kan identificeren is: verwerken strafuitspraak; er is gewoonweg geen ander proces dat op de stores (met name de store strafgegevens) het gewenste effect heeft. Maar voor die identificatie is er een *trigger* dataflow nodig naar dat proces (omdat het proces niet op eigen houtje kan beslissen dat de tijd is verstreken). Die trigger kan alleen komen van zoiets als een klok-proces. Laten we een proces KLOK invoeren en de dataflow van KLOK naar dat proces 'tijd' noemen. De SD kunnen we nu formeel in overeenstemming brengen met deze nieuwe DFD door als transitie te kiezen:

E: aankomst van: tijd
C: de aangekomen data voldoet aan: 3 mnd later dan de aanvang van de schorsing
A: verwerken strafuitspraak.

Achteraf bezien vind ik dit een zinvolle verbetering van het DFD.

Tot slot De gesuggereerde begrippen en verbanden zijn geformuleerd in termen van een gesuggereerde semantiek van DFDs en SDs. Een formele semantiek is mij nog niet bekend, maar lijkt me —in eenvoudige gevallen— niet zo moeilijk te geven. Bijvoorbeeld, voor DFDs: iedere dataflow-lijn in een DFD is een datatype (van een beperkte soort), en ieder proces is een functie (met als argumenttype: lijsten van de inkomende dataflows, etc). Een store is dan een "functie met interne toestand", dat wil zeggen, een bepaald soort functie (die makkelijk met recursie en een extra parameter te specificeren is). Kortom, een DFD is niet veel anders dan de type-informatie van een functioneel programma van een bepaalde vorm.

Een iets ingewikkelder semantiek wordt verkregen door een DFD op te vatten als aftreksel van een CSP programma van een bepaalde vorm.

Kennis van 'de semantiek van DFDs en SDs' is niet alleen nodig om de begrippen en beweringen in het verband tussen DFDs en SDs te formaliseren, maar is mijns inziens ook nodig om gevoel te hebben of te krijgen voor de "elegantie" en "doeltreffendheid" van een gegeven DFD en SD.

Constructie van contextdiagrammen

Maarten Fokkinga

Versie van 19 september 1994

Inleiding Bij het vak Informatiesystemen (versie van september 1994) wordt gevraagd een contextdiagram (CD) te maken voor verscheidene casussen. Hieronder volgt een manier waarop het CD geconstrueerd zou kunnen worden. Ik neem de casus Watersportbedrijf als voorbeeld.

Deze notitie is bedoeld als informeel stuk voor verbetering van mijn eigen begrip. Ik ontvang graag suggesties voor verdere verbetering.

* * *

Stap 1: entiteiten Ik begin met het onderscheiden van de relevant geachte externe (en de enige interne) entiteiten. Hiervoor heb ik geen interessante methode. Voor de casus kom ik in eerste instantie tot:

A (administratie, de interne entiteit), H (huurder), B (bedrijfsleiding).

Dit lijstje is niet volledig (maar dat weet ik nu nog niet).

Stap 2: interacties inventariseren Nu ga ik de tekst doorlezen, en per zin noteren of er een interactie is tussen de onderscheiden entiteiten. Voor de casus levert dat het volgende lijstje. De volgorde en nummering doet niet ter zake; die is er alleen voor de verwijzingen. Een toelichting volgt nog na het lijstje.

1. $H \leftrightarrow A$ aanvraag plus huurgegevens (voor '$\leftrightarrow$' zie onder)
2. $A \rightarrow H$ overzicht beschikbare jachten
3. $H \rightarrow A$ jachtkeuze
4. $A \rightarrow H$ rekening
5. $H \rightarrow A$ aanbetaling
6. $A \rightarrow H$ definitief/ongedaan maken contract
7. $A \rightarrow B$ transportlijst
8. $? \rightarrow A$ aankomstmelding (= aanvullende huurgegevens) (voor '?' zie onder)
9. $A \rightarrow H$ definitieve rekening
10. $H \rightarrow A$ eindbetaling (of niet)
11. $A \rightarrow H$ 1ste herinnering
12. $A \rightarrow H$ 2de herinnering
13. $A \rightarrow I$ relevante huurgegevens (voor 'I' zie onder).

De pijlen $\rightarrow$ en $\leftrightarrow$ duiden een richting aan waarin de data (if any) stroomt. Bij sommige interacties is er sprake van een "onderhandeling" in plaats van een eenrichtingsstroom van data; zie interactie 1. Dat is niet geoorloofd in DFDs en CDs. Kennelijk heb ik interactie 1 nog op een te hoog abstractienivo ge-identificeerd. Om tot uitsluitend $\rightarrow$ pijlen te komen, moet ik die hoog-nivo interactie verder verfijnen, of een richting (al of niet terecht) opleggen. Ik kies bij interactie 1 voor dat laatste. Voor het gemak heb ik verder geen 'onderhandelingsinteracties' meer opgevoerd, maar ze al direct tot eenrichtingsinteracties verfijnd.

Bij interactie 8 staat een vraagteken; het is mij niet duidelijk waar de data vandaan komt. Op dit punt aangekomen dienen we hetzij een nieuwe entiteit te bedenken, hetzij (al of niet terecht) de huurder H als data-bron te nemen. Dat laatste doe ik (om aan te sluiten bij de standaard uitwerking).

Bij interactie 13 blijkt dat ik de entiteit I (incasso) nodig heb; dat had ik tevoren niet gezien. Ik voeg I (incasso) toe aan het lijstje van externe entiteiten.

Stap 3: data onderscheiden Nu ga ik bij iedere interactie bepalen of er sprake is van data, en ga ik de soorten data onderscheiden. Bij interacties 1–5 is er sprake van data, en wel: verschillende datastromen. Bij interactie 6 kies ik ervoor om hier niet van data te spreken (om overeenstemming te bereiken met de standaard uitwerking). Regel 6 zul je verederop dus niet meer tegenkomen. Bij interactie 9 (de definitieve rekening) vind ik dat er weliswaar sprake is van data, maar die kan ik van dezelfde soort beschouwen als de data bij interactie 4 (de voorlopige rekening); de standaard uitwerking onderscheidt ze. Bij interactie 10 (de eindbetaling) kan ik ook weer de data als 'van dezelfde soort' verklaren als de data bij interactie 5 (de aanbetaling); dit gebeurt ook in de standaard uitwerking. Etcetera, etcetera.

Ikzelf ben ook geneigd om de verschillende herinneringen als dezelfde soort data te beschouwen. De standaard uitwerking maakt er onderscheid tussen.

Stap 4: Vervolledigen Vervolgens ga ik na of er voor iedere dataflow uit het systeem A een dataflow naar het systeem is aan te wijzen die als trigger optreedt. Bij ontbreken daarvan voeg ik zo'n trigger-dataflow toe en zonodig een externe entiteit waar die vandaan komt, bijvoorbeeld een klok K. Dit doe ik omdat Rob van de Weg dat wil. Dit leidt tot toevoeging van de volgende interacties:

11'.	$K \rightarrow A$	trigger voor interactie 11 (1ste herinnering naar Huurder)
12'.	$K \rightarrow A$	trigger voor interactie 12 (2de herinnering naar Huurder)
13'.	$K \rightarrow A$	trigger voor interactie 13 (relevante huurgegevens naar Incasso).

Ook hier kan ik er weer voor kiezen om de soorten data als dezelfde te beschouwen, of om ze te onderscheiden. Ik identificeer ze.

Tenslotte kijk ik of alle "uitgaande interacties" door het systeem genomen kunnen worden op grond van de in het systeem binnengekomen data. (Het is waarschijnlijk dat deze handeling bij het maken van een gedetaileerder DFD meer effect heeft dan nu, bij het CD.)

Het blijkt dat ik vergeten ben om de initiele jacht- en chartergegevens in het administratiesysteem A te laten binnenstromen. Dus voeg ik toe:

0. B → A initiele jacht- en chartergegevens.

Stap 5: Het diagram Al met al heb ik nu het volgende lijstje:

0.	B→A	initiele jacht- en chartergegevens	
1.	A←H	aanvraag plus huurgegevens	
2.	A→H	overzicht beschikbare jachten	
3.	A→H	jachtkeuze	
4,9.	A→H	rekening	[eerste en definitieve]
5,10.	A←H	betaling	[aan-, eind-]
7.	A→H	transportlijst	
8.	A←H	aankomstmelding	
11',12',13'.	K→A	trigger	[voor 11, 12, en 13]
11,12.	A→H	herinnering	[de 1ste, en 2de]
13.	A→I	relevante huurgegevens.	

Wanneer ik nu een tekening maak van bovenstaand lijstje (door één langgerekt bolletje te tekenen om alle As, en net zo voor alle Hs, etc), dan heb ik een contextdiagram. De opmaak van de richtingen van de interacties heb ik zó gekozen dat de bolletjes mooi getekend kunnen worden. Het diagram komt aardig (maar niet helemaal) overeen met de standaard uitwerking. Het past bovendien op 11 regels van een A4-tje.

* * *

Tot slot Het stappenplan lijkt algemener toepasbaar dan alleen de casus Watersportbedrijf. In bovenstaande uitwerking heb ik echter nog twee problemen, waarvoor ik geen oplossing heb. Ten eerste: wat zijn nou eigenlijk 'interacties'? (Ik heb zorgvuldig het woord 'actie' vermeden, omdat dat al een betekenis krijgt bij statusdiagrammen.) En ten tweede: waarom is er bij interactie 6 (het definitief of ongedaan maken van het contract) geen sprake van data, volgens de standaard uitwerking?