Capitolul 11

Proiectarea sistemelor informatice

Etape

- Etapele clasice ale dezvoltarii unui sistem informatic (SI) sunt urmatoarele:

1. Studiul de fazabilitate
2. Analiza de sistem
3. Proiectarea
4. Codificarea
5. Testarea integrarii
6. Implementarea

- Pentru fiecare din aceste etape au fost dezvoltate diverse unelte si metodologii.

Obiectiv

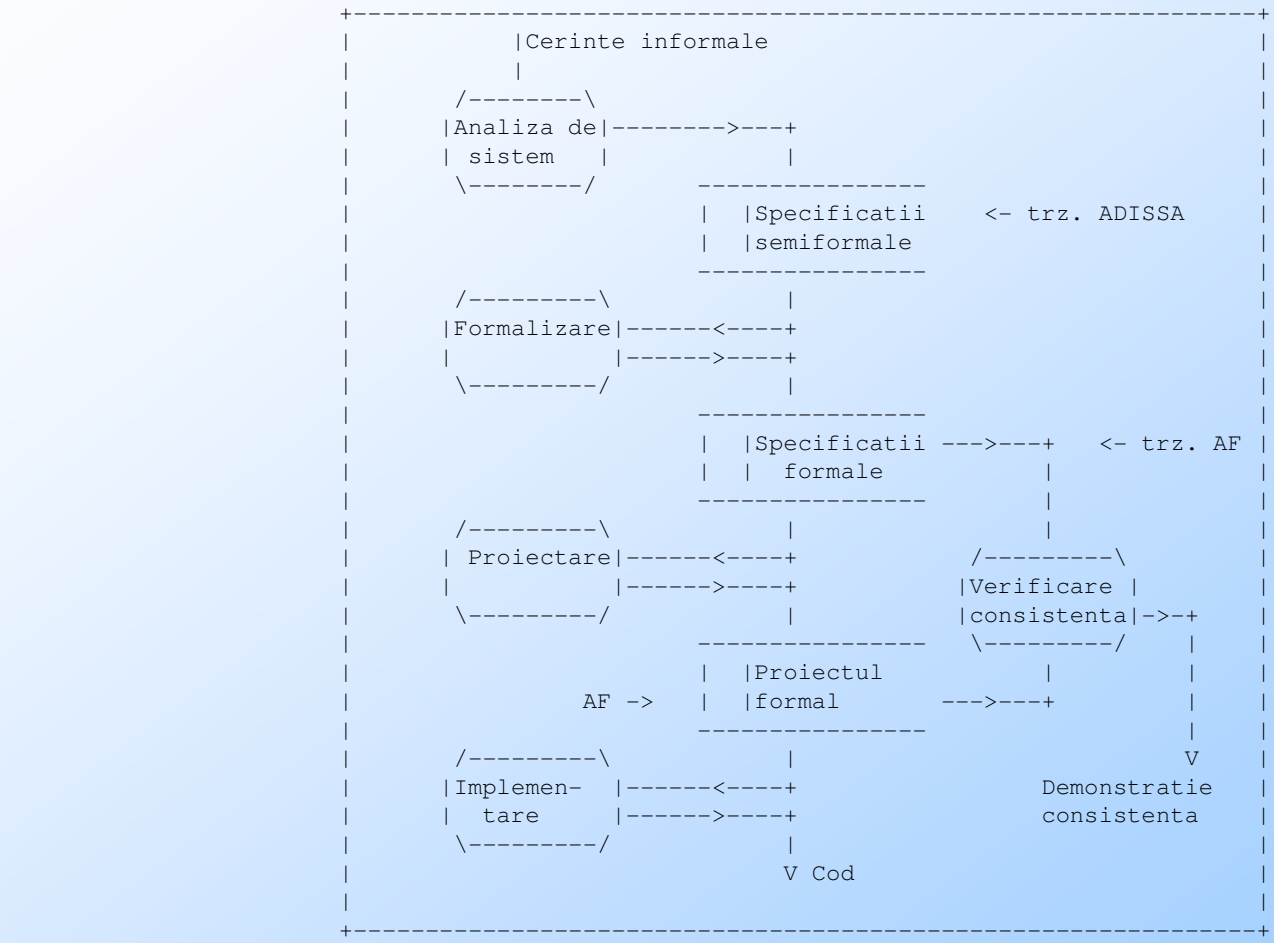
- In prezentarea de fata se va prezenta metoda ADISSA (Architectural Design of Information Systems based on Structured Analysis) care este o extensie a SSA (Structured Systems Analysis).
- Se poate demonstra matematic consistenta intre specificatia si proiectul partii de control a tranzactiilor ADISSA.
- Pentru modelarea acestor tranzactii se folosesc automatele finite (AF) din cauza faptului ca:

1. AF sunt bine definite matematic, permitand demonstrarea consistentei.
2. SI interactive sunt usor de modelat si realizat folosind AF.
3. Codul pentru un AF este scurt si simplu.

Cuprins

- O trecere in revista a procesului de dezvoltare a unui SI pe baza metodologiei prezentate.
- Prezentarea metodei ADISSA. Este introdus aici conceptul de tranzactie (in sens ADISSA).
- Formalizarea specificarii tranzactiilor, introducandu-se conceptul de tranzactie AF si reguli de transformare a tranzactiilor ADISSA in tranzactii AF.
- Etapa de proiectare. Sunt introduse automatele finite si algoritmi de realizare a lor.
- Teoremele mai importante care duc la demonstrarea consistentei intre specificatia formala a unei tranzactii si realizarea sa prin AF.
- In final unele asertiuni si restrictii legate de metoda prezentata

Etape de dezvoltare SI



Descriere etape

- Primul pas al dezvoltarii (Etapa 1) este analiza de sistem, care se va formaliza cu ajutorul metodei ADISSA. Aceasta metoda este orientata catre specificarea prelucrarilor in sisteme informatice interactive.
- Conceptul de tranzactie care se va introduce aici semnifica o prelucrare independenta definita de utilizator. Rezultatele aplicarii ADISSA sunt specificatiile semi-formale ale SI.
- Ele nu au un caracter total formal deoarece pentru fiecare tranzactie se pot da mai multe interpretari, inclusiv unele care nu sunt necesare utilizatorului SI.

Descriere etape

- Urmeaza formalizarea (Etapa 2), bazata pe abordarea transformationala. Cuprinde selectarea interpretarilor relevante pentru fiecare tranzactie si descrierea metodei folosite pentru obtinerea tranzactiilor AF.
- Aici vor fi prezentate regulile pe baza carora o tranzactie ADISSA se transforma intr-o tranzactie AF.

Descriere etape

- Pasul al treilea (Etapa 3), proiectarea, arata cum se implementeaza o tranzactie AF prin intermediul unei multimi de automate finite.
- O tranzactie AF este descompusa in mai multe subtranzactii simple, continand elementele tranzactiei initiale la care se adauga functii "administrative" pentru conectarea subtranzactiilor.

Descriere etape

- Fiecare subtranzactie este implementata printr-un AF.
- Verificarea consistentei (Etapa 4) intre o tranzactie AF si implementarea sa prin AF este facuta folosind asertiunile pasilor precedenti si un set de teoreme.

Etapa 1: ADISSA

- Metoda ADISSA este o rafinare a metodei SSA si utilizeaza diagrame de flux de data (DFD) nu numai pentru analiza de sistem ci si pentru proiectarea acestuia.
- Rezultatele principale ale aplicarii ei sunt:

1. Interfata utilizator, formata dintr-un arbore de meniuri care poate fi considerat ca fiind arhitectura externa a sistemului.
2. Arhitectura interna, formata dintr-un set de tranzactii activate ca urmare a diferitelor evenimente sau a unor cereri ale utilizatorilor.

- Pe linga acestea, metoda produce si schema bazei de date si schema de intrari iesiri ale SI.

Etapa 1: Pasi

- Pasii principali ai aplicarii ADISSA sunt:

1. Analiza functionala si constructia DFD
2. Identificarea tranzactiilor
3. Proiectarea arborelui de meniuri
4. Descrierea structurala a tranzactiilor
5. Proiectarea schemei bazei de date
6. Proiectarea schemei de intrari - iesiri

Analiza functionala si constructia DFD

- Analiza functionala este bazata pe folosirea unei ierarhii de diagrame de flux de date (DFD). Acestea definesc si descriu:

1. ***Functiile*** (prelucrarile) efectuate de sistem
2. ***Datele memorate*** care formeaza suportul permanent de date al sistemului.
3. ***Entitatile externe*** care se interfateaza cu sistemul
4. ***Fluxurile de date*** (FD) care conecteaza elementele de mai sus.

Functii

- ***Functiile*** specifica prelucrari efectuate in sistem. Ele sunt de doua feluri: ***functiile generale*** sunt acelea care pot fi detaliate ulterior prin DFD asociate lor; ***functiile elementare*** specifica prelucrari elementare care nu se pot detalia prin DFD.
- ***Reprezentarea grafica*** normala pentru functii este cercul: dublu pentru functiile generale si simplu pentru cele elementare.
- Pe parcursul prezentarii de fata, in interiorul ovalului unei functii acestora este inclus un identificator al functiei (o secventa de numere care deriva din ierarhia de DFD), ele fiind explicitate in legenda asociata figurilor. Codul lor este de tip FGi sau FEj.

Entitati externe

- ***Entitatile externe*** (numite in continuare entitati) specifica elementele externe cu care se interfateaza sistemul informatic si care reprezinta declansatori si terminatori ai tranzactiilor.
- Entitatile sunt de trei feluri:

1. ***Entitate utilizator (EU)*** specifica interfata SI cu utilizatorii (umani) ai acestuia. Ele modeleaza tipuri de utilizatori (analog cu entitatile din modelul ELE, descris in partea a doua a cursului).

   Sunt atit ***declansatori*** de tranzactii (cand utilizatorul lanseaza o optiune a sistemului de meniuri al SI) cit si ***terminatori*** de tranzactii (cand datele rezultate din prelucrari sunt destinate prezentarii catre aceea categorie de utilizatori).

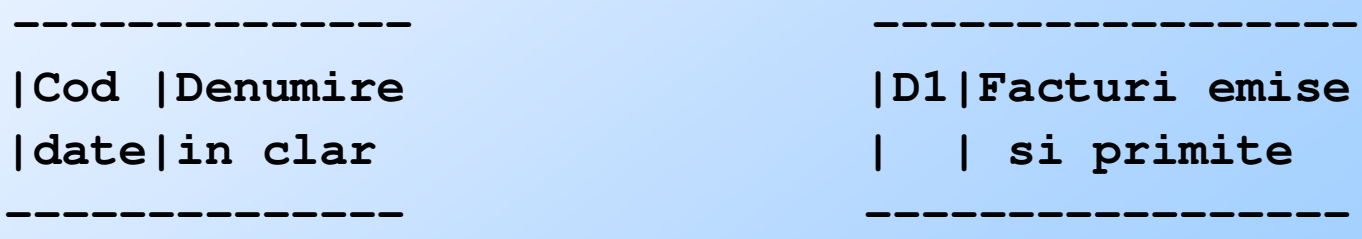
Entitati externe

Reprezentarea grafica va fi un dreptunghi in care este scris codul acelei entitati (ex. EU3), explicitarea lor facandu-se in legenda atasata figurii. Ele sunt prezentate in exteriorul casetei ce inconjoara portiunea DFD care modeleaza prelucrarile din SI (deci nu se confunda cu functiile, care sunt in interiorul acesteia si nu au in codificare sirul 'EU').

2. ***Entitate timp (ET)*** modeleaza activarea unor tranzactii la momente de timp prestabilite (exemplu: sfirsit de zi, sfirsit de luna, sfirsit de an, etc).
3. ***Reprezentarea grafica*** va fi un triunghi in care este inscris codul entitatii (ex. ET sau ET1).

Date memorate

- ***Datele memorate (DM)*** modeleaza partea "statica" a SI, suportul de date care este actualizat sau consultat de catre functiile prezente in DFD.
- Fiecare astfel de element al diagramei se poate implementa conceptual si fizic prin una sau mai multe relatii (fisiere de date) utilizate de SI.
- Reprezentarea grafica este urmatoarea:



Flux de date

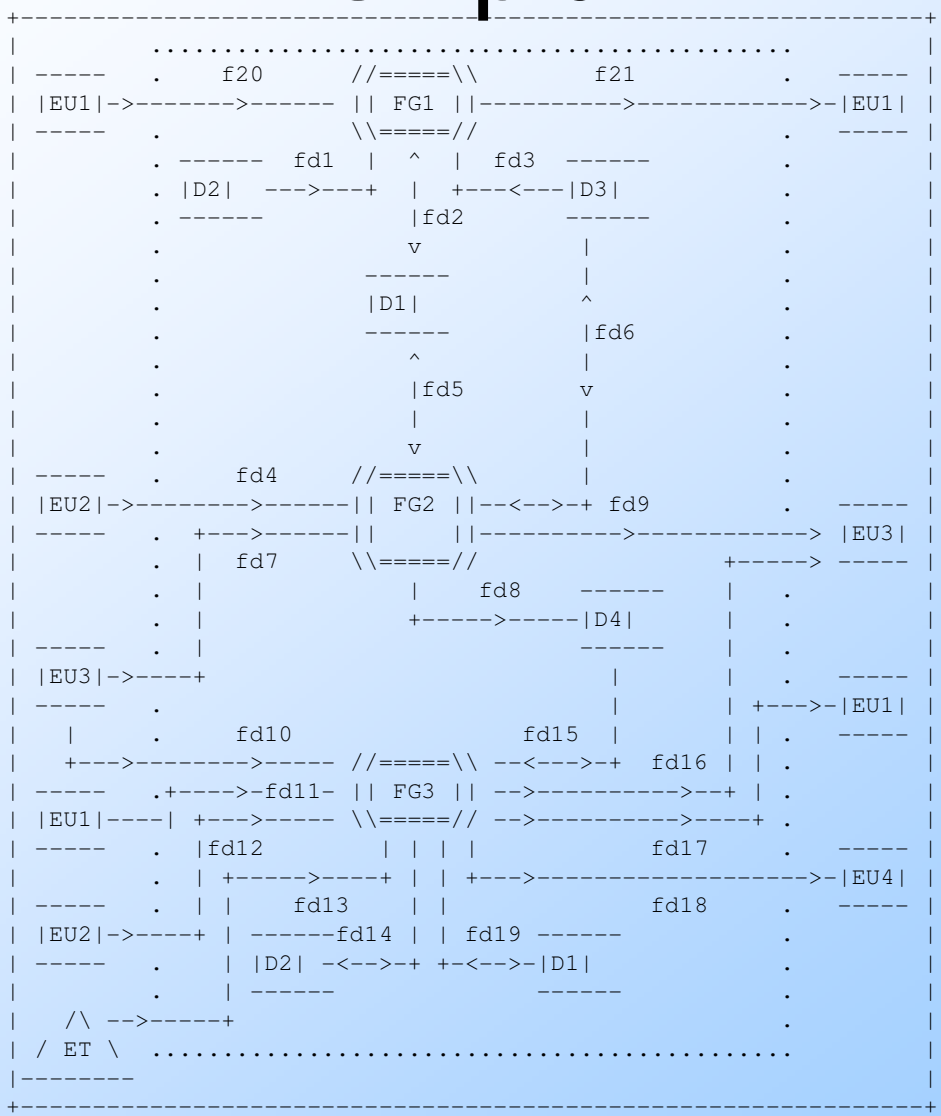
- Fluxurile de date modeleaza circulatia datelor intre entitati, functii si date memorate.
- Ele se reprezinta grafic prin arce orientate in sensul de circulatie al datelor.
- Observatie: Daca intr-o DFD este prezenta de mai multe ori acelasi element (deci cu acelasi cod), el reprezinta aceeasi categorie de obiecte in ipostaze diferite.

Tranzactie ADISSA

- Conceptul de tranzactie (ADISSA) este un element de baza al metodei si poate fi definit din doua puncte de vedere:

1. ***Din punct de vedere al utilizatorului***, o tranzactie este o prelucrare independenta care:
   a. are un inceput (start)
   b. efectueaza o prelucrare definita de utilizator
   c. are un sfirsit (stop, terminare) care lasa sistemul intr-o stare coerenta (in sensul in care este definita coerenta in teoria bazelor de date)
2. ***Din punct de vedere al DFD***, o tranzactie consta in:
   a. una sau mai multe entitati care declanseaza tranzactia
   b. una sau mai multe functii elementare legate intre ele prin fluxuri de date
   c. toate celelalte elemente (entitati si date memorate) conectate cu aceste functii si dintre care o parte formeaza terminatorii acelei tranzactii.

Exemplul 1


EU1
EU2
EU3
EU4
ET
FG1
FG2
FG3
D1
D2
D3
D4
f20
f21
fd1
fd2
fd3
fd4
fd5
fd6
fd7
fd8
fd9
fd10
fd11
fd12
fd13
fd14
fd15
fd16
fd17
fd18
fd19

Exemplul 1 – cont.

Entitati:

EU1: Furnizori
EU2: Constructori
EU3: Clienti
EU4: Contabili
ET: Calendar de evenimente

Functii:

FG1: Aprovizionare de la furnizori
FG2: Cereri de lucrari si ordine de lucrari
FG3: Inregistrari contabile si rapoarte

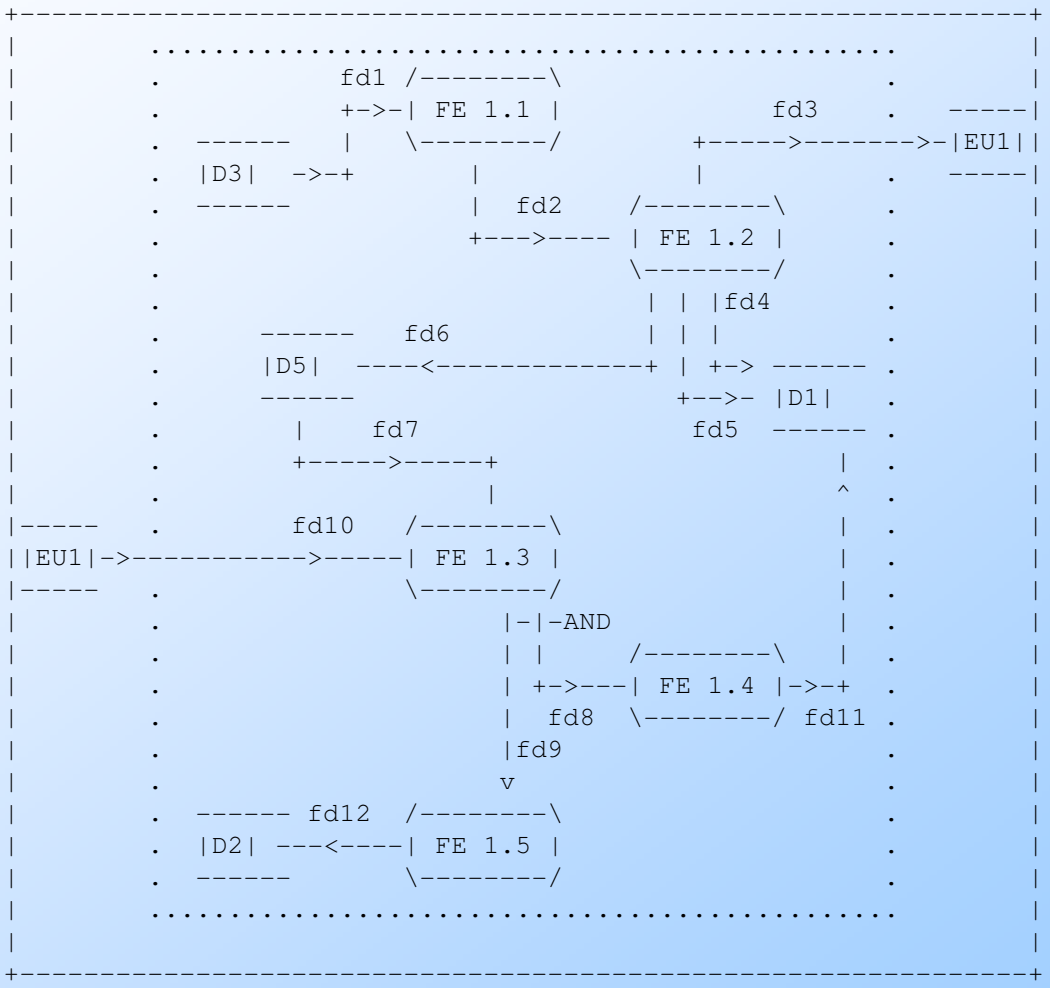
Date memorate:

D1: Materiale in stoc
D2: Conturi furnizori
D3: Dereri de lucrari si ordine de lucrari
D4: Conturi clienti

Fluxuri de date:

fd1, fd20: Facturi si receptii
fd2: Stoc
fd3: Ordine in lucru
fd4: Materiale necesare
fd5: Preturi mater.din stoc
fd6: Acceptare/refuz
fd7: Acceptare/refuz
fd8: Detalii client, preturi
fd9: Deviz propus (pt.acont)
fd10: Incasari
fd11: Adrese de la furnizori
fd12: Costuri
fd13: Sfirsit de luna
fd14: Plati, balante/furnizori
fd15: Plati, balante/clienti
fd16: Receptii,adrese,facturi
fd17: Plati de efectuat
fd18: Rapoarte
fd19: Tranzactii
fd21: Ordine de cumparare

Exemplul 2: Detaliere FG1



Exemplul 2 – cont.

Entitati: EU1: Furnizori

Date memorate:

D1: Materiale in stoc
D2: Conturi furnizori
D3: Dereri de lucrari si ordine de lucrari
D5: Ordine de cumparare

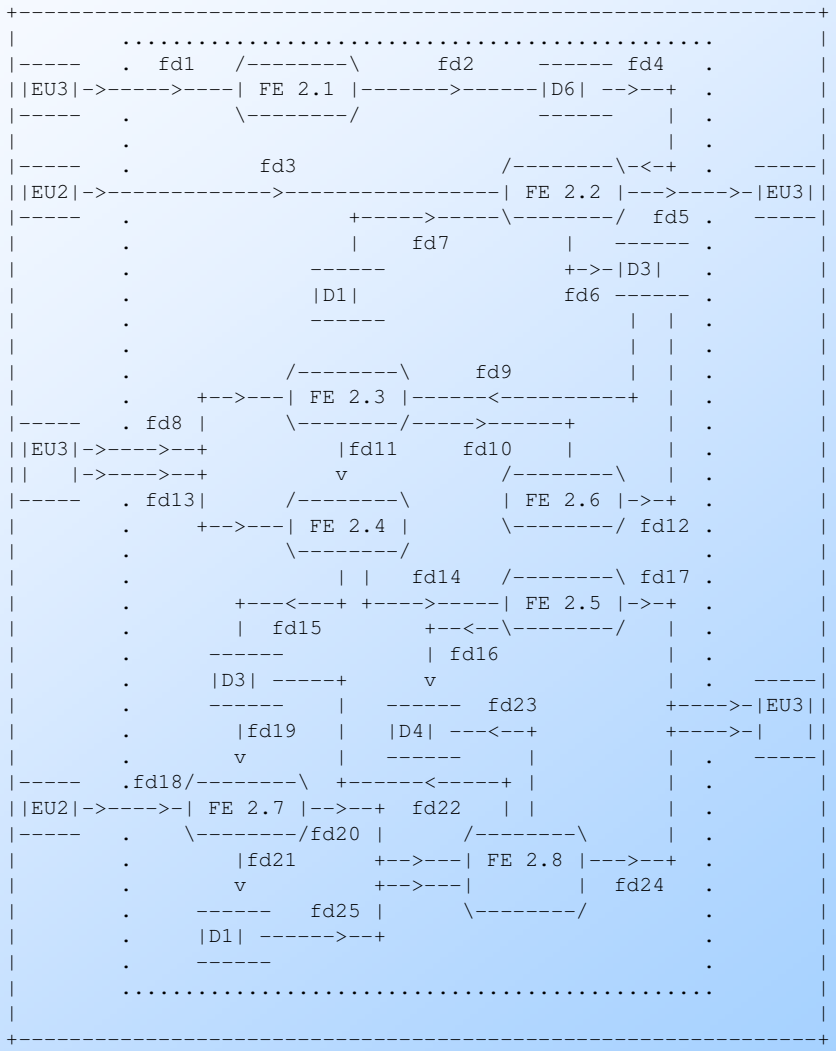
Functii:

FE1.1: Cautare cereri de lucrari
FE1.2: Pregatire lista de materiale necesare
FE1.3: Inregistrare factura primita (de la furnizor)
FE1.4: Actualizare stoc
FE1.5: Actualizare cont furnizor

Fluxuri de date:

fd1: Ordine in lucru
fd2: Detalii ordine active
fd3: Ordine de cumparare
fd4: Materiale comandate
fd5: Stoc
fd6: Ordine de cumparare
fd7: Ordine de cumparare
fd8: Materiale cumparate
fd9: Facturi si receptii
fd10: Facturi si receptii
fd11: Materiale cumparate
fd12: Facturi

Exemplul 3: Detaliere FG2



Exemplul 3 – cont.

Entitati: cu aceeasi semnificatie ca la DFD-0
Date memorate:
D1, D2, D3, D4:cu aceeasi semnificatie ca la DFD-0
D6: Cereri de lucrari
Functii:
FE2.1: Inregistrare cerere de lucrari
FE2.2: Pregatire deviz
FE2.3: Inregistrare raspuns client
FE2.4: Inregistrare prima plata
FE2.5: Deschidere cont client
FE2.6: Inchidere lucrare
FE2.7: Inregistrare consumuri
FE2.8: Actualizari si rapoarte
Fluxuri de date:

fd1: Cereri de lucrari	fd13: Prima plata
fd2: Cereri de lucrari	fd14: Detalii plata
fd3: Necesar de materiale	fd15: Acceptare
fd4: Cereri de lucrari	fd16: Detalii plata
fd5: Propuneri lucrari (deviz)	fd17: Factura
fd6: Propuneri de lucrari	fd18: Consumuri
fd7: Materiale si preturi	fd19: Detalii lucrare
fd8: Acceptare/refuz	fd20: Materiale cons.+det.lucrare
fd9: Propuneri de lucrari	fd21: Materiale consumate
fd10: Refuz	fd22: Sfirsit lucrare
fd11: Acceptare	fd23: Cost lucrare
fd12: Propuneri de sters	fd24: Rapoarte de lucru

fd25: Materiale si preturi

Etapa 1: Pasi

- Pasii principali ai aplicarii ADISSA sunt:

1. Analiza functionala si constructia DFD
2. ***Identificarea tranzactiilor***
3. Proiectarea arborelui de meniuri
4. Descrierea structurala a tranzactiilor
5. Proiectarea schemei bazei de date
6. Proiectarea schemei de intrari - iesiri

Identificare traanzactii

- Acest pas al metodei include identificarea tranzactiilor si a elementelor componente ale acestora, in principal a functiilor elementare si a datelor memorate continute.
- **Se face prin inspectarea DFD si impartirea lor in tranzactii, conform definitiei de mai sus (slide 18)**.

Identificare traanzactii

I. In DFD-1 (fig.3) pot fi identificate doua tranzactii:
   1. prima contine functiile 1.1 si 1.2, entitatea EU1 si datele memorate D1, D3 si D5.
   2. a doua contine functiile 1.3, 1.4 si 1.5 si toate celelalte elemente conectate la acestea.

II. In DFD-2 (fig.4) identificam patru tranzactii, care contin:
   1. functia 2.1
   2. functia 2.2
   3. functiile 2.3-2.6
   4. functiile 2.7, 2.8

+ celelalte elemente conectate la aceste functii

Identificare traanzactii

- DFD nu contin si semnificatia prelucrarilor efectuate de tranzactii. Aceasta se va inregistra separat.
- Deoarece DFD nu contin informatii despre fluxul controlului (ci doar al datelor), o tranzactie poate fi interpretata in mai multe feluri (are mai multe semnificatii posibile).
- Detectarea interpretarilor relevante (necesare utilizatorilor) va fi prezentata de asemenea in acest capitol.

Etapa 1: Pasi

- Pasii principali ai aplicarii ADISSA sunt:

1. Analiza functionala si constructia DFD
2. Identificarea tranzactiilor
3. ***Proiectarea arborelui de meniuri***
4. Descrierea structurala a tranzactiilor
5. Proiectarea schemei bazei de date
6. Proiectarea schemei de intrari - iesiri

Proiectarea arborelui de meniuri

- Scopul acestui pas este crearea unei ierarhii de meniuri care reprezinta, asa cum am mentionat, arhitectura esterna a SI.
- Aceste meniuri se deduc din DFD.
- Meniul initial consta din functiile conectate direct la entitatile utilizator si contin cite o linie pentru fiecare astfel de conexiune.
- Optiunile ("liniile") din fiecare meniu sunt de doua feluri:

Proiectarea arborelui de meniuri

a. linie terminala daca conexiunea este spre o functie elementara. Selectarea ei din meniu duce la activarea acestei functii. Acest tip de linii sunt frunzele arborelui de meniuri.
b. linie de selectie daca conexiunea este spre o functie generala. Selectarea ei duce la afisarea submeniului asociat acelei functii generale (dedus din DFD al acelei FG). Acest tip de linii sunt nodurile intermediare ale arborelui de meniuri.

- Rafinarea arborelui de meniuri se face astfel incit fiecare linie terminala sa poata fi atinsa printr-o cale care porneste din meniul initial si selectarea ei duce la declansarea unei tranzactii.

Etapa 1: Pasi

- Pasii principali ai aplicarii ADISSA sunt:

1. Analiza functionala si constructia DFD
2. Identificarea tranzactiilor
3. Proiectarea arborelui de meniuri
4. ***Descrierea structurala a tranzactiilor***
5. Proiectarea schemei bazei de date
6. Proiectarea schemei de intrari - iesiri

Descrierea structurala a tranzactiilor

- In metoda ADISSA, aceasta descriere se face la doua nivele:
- La nivelul superior, fiecare tranzactie este descrisa specificand:
  - elementele care compun tranzactia (functii, entitati, date memorate si fluxurile de date care le conecteaza)
  - fluxul controlului, care poate fi opus fluxului datelor.
- Specificarea fluxului de control se face in limbaj natural (vom vedea mai tarziu ca in momentul transformarii in tranzactii AF fluxul controlului se va putea specifica pe diagrama)

Descrierea structurala a tranzactiilor

- La nivelul inferior sunt detaliate componentele tranzactiei intr-o forma structurata (de exemplu functiile se pot detalia folosind pseudocodul)
- ***Observatie:*** descrierea unei tranzactii (asa cum este definita in acest subcapitol) este exprimarea uneia sau mai multor interpretari valide ale semanticii sale. Aceasta descriere nu este suficient de formala pentru o transformare automata (de altfel am mentionat ca rezultatul analizei functionale sunt niste specificatii semiformale)

Etapa 1: Pasi

- Pasii principali ai aplicarii ADISSA sunt:

1. Analiza functionala si constructia DFD
2. Identificarea tranzactiilor
3. Proiectarea arborelui de meniuri
4. Descrierea structurala a tranzactiilor
5. ***Proiectarea schemei bazei de date***
6. Proiectarea schemei de intrari - iesiri

Proiectarea schemei bazei de date

- Se poate folosi orice metoda valida, inclusiv metoda bazata pe modelul entitate asociere (EA) prezentat anul trecut in cadrul cursului de Baze de date.
- Rezultatul acestui pas este schema bazei de date in forma normala dorita.
- Nu insistam asupra acestui pas, fiind tratat in extenso anterior.

Etapa 1: Pasi

- Pasii principali ai aplicarii ADISSA sunt:

1. Analiza functionala si constructia DFD
2. Identificarea tranzactiilor
3. Proiectarea arborelui de meniuri
4. Descrierea structurala a tranzactiilor
5. Proiectarea schemei bazei de date
6. ***Proiectarea schemei de intrari - iesiri***

Proiectarea schemei de intrari - iesiri

- Acest pas consta din specificarea tipului si dispozitivului pentru fiecare intrare si iesire in/din fiecare tranzactie.
- Rezultatul acestui pas contine in principal:
  1. machetele ecranelor de introducere a datelor
  2. machetele ecranelor de prezentare a datelor (vizualizare)
  3. machetele rapoartelor produse de catre SI.
- Toate acestea formeaza ceea ce se numeste schema de intrari-iesiri a sistemului informatic.

Etapa 2. Formalizare

- Aceasta etapa trateaza:

1. problema interpretarilor multiple ale unei tranzactii
2. definirea tranzactiilor AF
3. reguli de transformare a tranzactiilor ADISSA in tranzactii AF

Interpretari multiple

- Unul din lipsurile majore ale modelului ADISSA este lipsa posibilitatii indicarii explicite a fluxului controlului, deci a ordinii in care sunt activate elementele componente ale unei tranzactii.
- Fluxurile de date (FD) specifica circulatia datelor dar ele pot fi si semnale care duc la declansarea unor functii.
- DFD, asa cum au fost descrise pina acum, nu dau nici o indicatie privind care FD sunt si declansatori si, in cazul ca sunt, in ce directie. Din aceasta cauza, pentru fiecare FD pot fi date mai multe interpretari posibile.

Interpretari multiple

- Utilizatorul este cel care trebuie sa indice care FD sunt declansatori si in ce directie. Pentru aceasta, el trebuie sa descrie ce prelucrare executa fiecare tranzactie.
- Descrierea se face desemnand:

1. un element de pornire (start)
2. unul sau mai multe elemente de oprire (stop, terminare)
3. fluxul controlului care uneste elementul de start cu elementele de stop, trecand prin diverse functii elementare.

Interpretari multiple

- Fiecare astfel de descriere a unei prelucrari efectuate de o tranzactie se numeste ***interpretare relevanta*** a acelei tranzactii.

a. De mentionat ca pentru fiecare tranzactie ADISSA sunt posibile mai multe ***interpretari***.

b. Doar o parte a acestora ***au sens*** si

c. Doar o parte a celor cu sens sunt ***relevante*** (deci necesare utilizatorului si cerute de catre acesta).

- Doar acestea din urma vor fi cele descrise la acest pas si implementate mai departe.

Exemplu

- Pantru diagrama din exemplul 3 putem da urmatoarele interpretari
- ***Interpretarea nr. 1:*** Constructorul (EU2) introduce datele despre materialele consumate, functia FE 2.7 actualizeaza D1 (materiale in stoc) in functie de datele introduse si de detaliile lucrarii prelevate din D3 (propuneri de lucrari si ordine de lucrari). In continuare este activata functia FE 2.8 care actualizeaza D4 (conturi clienti) si tipareste un raport care va fi trimis clientului (EU3).
- ***Interpretarea nr. 2:*** Clientul (EU3) cere raportul de lucru. Pentru aceasta el activeaza functia FE 2.8 care, pentru a putea tipari raportul, are nevoie de datele pe care i le furnizeaza functia FE 2.7. Deci FE 2.8 va activa FE 2.7 care livreaza datele (materiale consumate si detalii lucrare - manopera, etc) dupa care tipareste raportul de lucru.

Exemplu

- ***Interpretarea nr. 3:*** Constructorul (EU2) introduce datele privind consumurile de materiale, activand FE 2.7. Aceasta actualizeaza D1 si cu aceasta prelucrarea ia sfirsit.
- Din aceste trei interpretari, utilizatorul decide de exemplu ca doar primele doua sunt reprezinta prelucrari pe care le doreste implementate de catre SI.
- Deci doar aceste doua interpretari sunt relevante in sensul definitiei de mai sus si doar ele vor fi formalizate utilizand tranzactii AF.

Etapa 2. Formalizare

- Aceasta etapa trateaza:

1. problema interpretarilor multiple ale unei tranzactii
2. ***definirea tranzactiilor AF***
3. reguli de transformare a tranzactiilor ADISSA in tranzactii AF

Tranzactii AF

- O tranzactie AF este o descriere formala a fluxului de control pentru o tranzactie ADISSA.
- Functiile elementare, entitatile si datele memorate, asa cum au fost definite in ADISSA, sunt componente si ale tranzactiei AF si vor fi numite in continuare ***obiecte***.
- Ambiguitatea privind fluxul controlului este ridicata introducand conceptul de ***flux de date si control (FDC).***

Tranzactii AF

- Un FDC este un FD care in acelasi timp este si declansator al unei functii elementare sau entitati (in ultimul caz indica sfirsitul unei tranzactii).
- Un FDC este orientat de la obiectul declansator la obiectul declansat.
- Acele FD care nu sunt si de control raman in continuare fluxuri de date, semnificand doar circulatia datelor si nu si a controlului.

Tranzactii AF

- Deci o tranzactie ADISSA contine:

Elemente comune cu tranzactiile ADISSA:

1. functii elementare
2. entitati externe
3. date memorate

Elemente specifice:

4. fluxuri de date
5. fluxuri de date si control

Sintaxa tranzactii AF

- Inainte de a defini ce este o tranzactie AF corecta (deci sintaxa unei tranzactii AF) trebuie sa definim doua din caracteristicile unei astfel de tranzactii:

a. ***Multimea de start*** (pornire, declansare) este formata din entitatile care actioneaza ca declansatori pentru acea tranzactie.

b. ***Multimea de stop*** (oprire, terminare) este formata din toate entitatile si/sau datele memorate care au cel putin un FDC care intra si nici unul care iese.

Definitia tranzactiei AF

- O multime de obiecte, FD si FDC formeaza o tranzactie AF daca este construita dintr-o tranzactie valida ADISSA si satisface urmatoarele restrictii:

1. Functiile si entitatile sunt legate prin FDC. Daca o functie sau entitate nu este conectata prin FDC de restul obiectelor, ea este inutila.
2. Orice functie poate fi atinsa prin cel putin o cale orientata formata din FDC, cale care porneste dintr-un element al multimii de start. In caz contrar, acea functie nu va fi niciodata activata in caz de executie a tranzactiei.
3. Orice cale formata din FDC trebuie sa se termine intr-un element al multimii de stop, altfel tranzactia are un ciclu infinit. De remarcat ca ciclurile sunt permise in masura in care ele demonstreaza ca au o cale de terminare a lor (in genul punctului fix in cazul recursivitatii).

Definitia tranzactiei AF

4. O entitate are cel mult un FDC care iese. Daca are mai multe, ar exista o ambiguitate privind care dintre aceste FDC va fi activat de catre ea.
5. FDC si FD nu pot lega mai mult de doua obiecte (fluxurile nu se divid in sine).
6. Doar FDC care ies dintr-o functie pot participa la un AND logic. In acest caz ele sunt activate impreuna de catre functia din care pleaca. Acesta este singurul tip de AND intre FDC care exista in cazul unei tranzactii AF.

- O tranzactie AF poate fi reprezentata printr-o diagrama formata din aceeasi simboli ca in cazul diagramelor ADISSA.

Semantica executiei unei trz. AF

- ***Asertiunea 1:*** Pentru fiecare functie elementara exista o *implementare exacta* a specificatiilor acelei functii.
- ***Asertiunea 2:*** Daca o entitate are un FDC care iese, ea va emite un semnal se activare in timp finit.
- In plus, se va presupune ca o tranzactie nu poate fi declansata de mai multe semnale simultan (eliminarea concurentei in declansarea tranzactiilor).
- Prezentam in cele ce urmeaza o serie de definitii care duc la o definire formala a executiei unei tranzactii AF.

Semantica executiei - obiect

Definitia 1. *Executia unui obiect* = Este in functie de tipul obiectului:

a. ***Executia unei functii***: executia implementarii asociate acesteia (cf primei asertiuni, aceasta exista).

b. ***Executia unei entitati***: consta in emiterea unui semnal pe FDC care iese (daca are) sau emiterea unui semnal vid/sfarsit de tranzactie (daca nu are FDC care iese).

c. ***Executia unor date memorate***: emiterea unui semnal vid. Executia unor DM inseamna actualizarea/citirea acestora/accces la date - dupa care se emite un semnal vid dupa care fluxul se intrerupe. Semnalul vid este introdus pentru a respecta principiul ca executia fiecarui obiect duce la emiterea unui semnal, chiar si vid.

Semantica executiei - fluxuri

Definitia 2. *Semantica FDC:* Daca mai multe FDC care ies dintr-o functie nu sunt legate prin AND logic, functia activeaza doar pe unul dintre ele (deci se considera automat SAU EXCLUSIV - XOR).

- Daca sunt legate cu AND, functia le activeaza concurent pe toate dupa terminarea executiei sale.

Semantica executiei - eveniment

Definitia 3. *Eveniment:* Un eveniment este executia unui obiect al unei tranzactii AF, impreuna cu intrarile si iesirile sale.

Deci executia unui obiect (sau eveniment) este un triplet:

(i, A, r)

unde:

1. **i** = FDC care intra in acel obiect
2. **A** = executia acelui obiect (actiune)
3. **r** = multimea FDC care ies din obiect si sunt activate dupa executia sa in cazul intrarii **i** si executiei actiunii **A**.

- Daca A activeaza un singur FDC, **r** va contine un singur element.
- In cazul in care **A** activeaza mai multe FDC conectate printr-un AND, **r** va contine mai multe elemente.

Semantica executiei – ev. initial

- **Definitia 3.** *Eveniment initial* : Se spune ca E = (i, A, r) este un eveniment initial daca:

a. obiectul corespunzator lui **A** este un element din multimea de start (deci o entitate).

b. **i** simbolizeaza activarea acestei entitati de catre o cauza externa tranzactiei (deci **i** este un fel de pseudo-FDC. Multimea pseudo-FDC se va nota in cele ce urmeaza cu λ).

Semantica executiei - activare

- **Definitia 4**. *Activarea unui eveniment* : Se spune ca evenimentul E1 = (i1, A1, r1) este activat de evenimentul E = (i, A, r) (notatie: E $\rightarrow$ E1) daca si numai daca i1 apartine lui r.
- In cazul unui eveniment initial, se spune ca acesta este activat de un eveniment extern.
- Ca un corolar, se spune ca evenimentul E = (i, A, r) activeaza evenimentele

  E1 = (i1, A1, r1), ..., En = (in, An, rn) daca si numai daca r = i1 $\cup$ i2 $\cup$ ... $\cup$ in.

Semantica executiei - executie

- **Definitia 5.** *Executia tranzactiei* : Un lant finit de evenimente (E1, E2, ...) activate de un eveniment initial (E1→E2→E3..., unde E1 este un eveniment initial) se numeste *o executie a tranzactiei*.
- **Definitia 6.** *Implementarea tranzactiei* = Multimea tuturor executiilor unei tranzactii.

Etapa 2. Formalizare

- Aceasta etapa trateaza:

1. problema interpretarilor multiple ale unei tranzactii
2. definirea tranzactiilor AF
3. ***reguli de transformare a tranzactiilor ADISSA in tranzactii AF***

Transformarea

- Aceasta transformare se face in doi pasi:

1. Definirea interpretarilor relevante
2. Ajustari pentru respectarea sintaxei tranzactiilor AF descrisi in continuare.

Definirea interpretarilor relevante

- Pentru fiecare interpretare relevanta a unei tranzactii, analistul construieste o diagrama continand elementele diagramei tranzactiei ADISSA care iau parte la acea interpretare. Pentru aceasta, el trebuie sa executa urmatoarele operatii:

1. Definirea multimii de start.
2. Definirea multimii de stop.
3. Marcarea explicita a relatiei dintre fluxuri (AND, OR, XOR).
4. Specificarea tipului fiecarui flux (FD sau FDC)
5. Introducerea de FDC aditionale cand fluxul controlului este opus fluxului datelor.

Regulile 1-5

Pentru realizarea acestor operatii, se folosesc regulile urmatoare:

Regula 1. ***Multimea de start:*** Daca o entitate declanseaza tranzactia, ea apartine multimii de start.

Regula 2. ***Multimea de stop:*** Daca o tranzactie se termina prin activarea unei entitati sau a unor date memorate, entitatea sau datele memorate apartin multimii de stop.

Regula 3. ***Relatia dintre fluxuri:***

a. Daca, cf. afirmatiilor utilizatorului, mai multe fluxuri care ies dintr-un obiect sunt intotdeauna activate impreuna, ele vor fi marcate ca legate printr-un AND.

b. Daca ele sunt activate impreuna doar in unele executii, vor fi marcate ca legate prin OR.

c. Daca in toate executiile doar unul este activat (nu neaparat acelasi, bineinteles), ele vor fi marcate ca legate prin XOR.

Regulile 1-5

Regula 4. ***Clasificarea in FD si FDC:***

a. Orice flux intre doua functii sau intre o functie si o entitate este FDC.

b. Daca un flux este de la o functie la date memorate, atunci:
   - daca nu e implicata in nici un AND, este FDC.
   - daca este implicata intr-un AND doar cu fluxuri care pleaca spre date memorate, este FDC

c. Toate celelalte fluxuri sunt FD.

Observatie:

- Daca dupa aceste transformari un AND contine doar FD, este sters marcajul acestei legaturi (nu si fluxurile!); in cazul in care contine si FD si FDC, AND-ul este modificat pentru a lega doar FDC (din el se elimima FD).
- Ca rezultat, AND-urile vor contine doar FDC.

Regula 5. ***FDC aditionale:*** Daca, cf.afirmatiilor utilizatorului, fluxul controlului este opus fluxului de date, se adauga un FDC in sensul fluxului de control.

Ajustari

- Urmare a aplicarii regulilor de mai sus, se obtine o diagrama pentru fiecare interpretare relevanta a tranzactiei ADISSA.
- In continuare trebuiesc efectuate doua activitati:

1. Punerea de acord cu sintaza tranzactiilor AF (definita anterior)
2. Fuzionarea acestor diagrame intr-una singura

- Aceste activitati se efectueaza folosind regulile urmatoare:

Regulile 6-10

Regula 6. ***Multiplicare entitati:*** Daca o entitate are mai mult de un FDC care iese, este multiplicata in tot atitea copii cite FDC ies din ea, fiecare mostenind unul dintre aceste fluxuri.

- In acest fel, cerinta de sintaxa conform careia o entitate poate avea doar un singur FDC care iese este indeplinita.
- De asemenea, daca are cel putin un FDC care iese si este si in multimea de stop, este multiplicata cu inca un exemplar suplimentar a.i. o copie a sa sa nu aiba nici un FDC care iese.
- In ambele cazuri, copiile pot mosteni toate FDC care intra si numele original.

Regulile 6-10

Regula 7. ***Fuzionare:*** Fuzionarea diagramelor (cite una pentru fiecare interpretare relevanta) intr-una singura se face astfel:

- *Datele memorate*: raman intr-un singur exemplar, colectand toate fluxurile (care intra si ies).
- *Functiile*: raman intr-un singur exemplar, colectand toate fluxurile (care intra si ies).
- *Entitatile*: vor fi deocamdata copiate din fiecare interpretare.
- *Fluxurile*: se copiaza cite un exemplar din fiecare flux.
- Daca in cel putin o interpretare este FDC, acesta va fi FDC si in diagrama fuzionata; celelalte raman FD. Fiecare flux va fi etichetat cu eticheta sa originala (toate copiile trebuie sa aiba aceeasi eticheta in diversele diagrame de interpretari relevante, inclusiv cele aditionale).

Regulile 6-10

Regula 8. ***Eliminare OR:*** Eliminarea lui OR se face prin descompunerea si multiplicarea fluxurilor in numarul de exemplare necesar expresiei rezultate.

Exemplu: daca fluxurile a, b, c sunt marcate ca formand un OR si:

a. in unele executii sunt activate toate trei
b. in alte executii doar b si c
c. in toate celelalte executii doar b

- Rezulta expresia:

  (a AND b AND c) XOR (b AND c) XOR b,

ceea ce inseamna ca avem nevoie de 1 copie a lui a, 3 copii ale lui b si doua copii ale lui c.

- OR intre cele trei fluxuri va fi deci inlocuit cu un AND intre 3 fluxuri, un AND intre alte doua fluxuri si un flux neconectat logic cu altele (obs.: XOR este implicit si nu se marcheaza pe diagrama)

Regulile 6-10

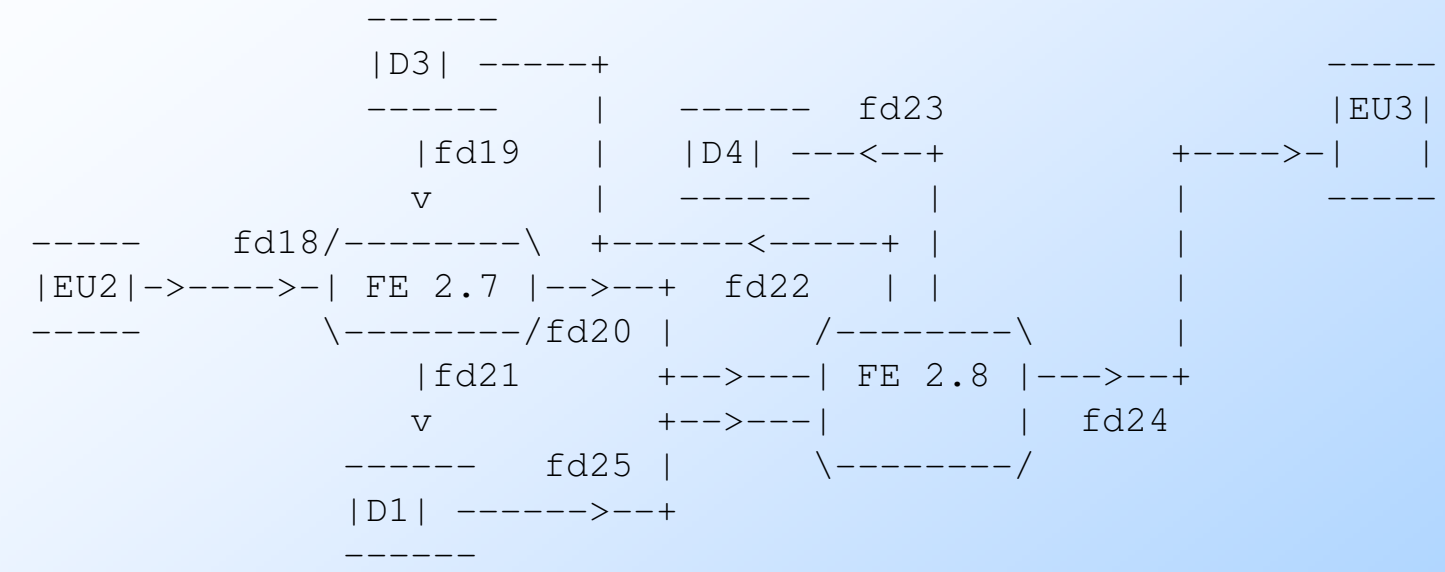
Regula 9. ***Fuziune entitati:*** Aparitiile multiple ale aceleiasi entitati (care pot fi rezultate si din multiplicare, vezi regula 6) pot fuziona daca au aceleasi FDC care ies, in care caz colecteaza toate fluxurile care intra.

Daca nu au aceleasi FDC care ies, raman distincte si se redenumesc, astfel incit sa nu avem doua entitati cu acelasi nume.

Regula 10. ***Specificare functii:*** Se face specificarea prelucrarilor efectuate de fiecare functie elementara (in orice mod: pseudocod, limbaj natural, etc).

Aceasta regula nu are nici o influenta in prezentarea metodologiei in cele ce urmeaza. Ea se foloseste doar cand aceasta se aplica pe un caz real.

Exemplu



FE2.7: Inregistrare consumuri
fd18: Consumuri
fd20: Materiale cons.+detalii lucrare
fd22: Sfirsit lucrare
fd24: Rapoarte de lucru

FE2.8: Actualizari si rapoarte
fd19: Detalii lucrare
fd21: Materiale consumate
fd23: Cost lucrare
fd25: Materiale si preturi

Interpretarea 1

Interpretarea nr. 1: Constructorul (EU2) introduce datele despre materialele consumate, functia FE 2.7 actualizeaza D1 (materiale in stoc) in functie de datele introduse si de detaliile lucrarii prelevate din D3 (propuneri de lucrari si ordine de lucrari). In continuare este activata functia FE 2.8 care actualizeaza D4 (conturi clienti) si tipareste un raport care va fi trimis clientului (EU3).

- **Regula 1:** Multimea de start: { EU2 }
- **Regula 2:** Multimea de stop: { EU3 }
- **Regula 3:** Fluxurile fd20 si fd21 care ies din FE2.7 sunt intotdeauna activate impreuna => se marcheaza cu AND. Analog fluxurile fd22, fd23, fd24 care ies din FE2.8.
- **Regula 4:** FDC vor fi fd18, fd20 si fd 24. Nici un fd de la functii spre date memorate nu e convertit in FDC, deoarece sunt in AND impreuna cu fluxuri care merg la entitati sau functii.
- Deoarece ambele AND-uri contin doar un FDC si in rest FD => prin restringerea AND-urilor conform observatiei, pentru a contine doar FDC, aceste AND-uri dispar (ramane cite un singur FDC in componenta lor). Rezulta deci ca in final aceasta diagrama nu contine AND.
- **Regula 5:** Nu este cazul pentru aceasta interpretare.

I1: Rezultat

Am simbolizat cu linie punctata FD si cu linie continua FDC.

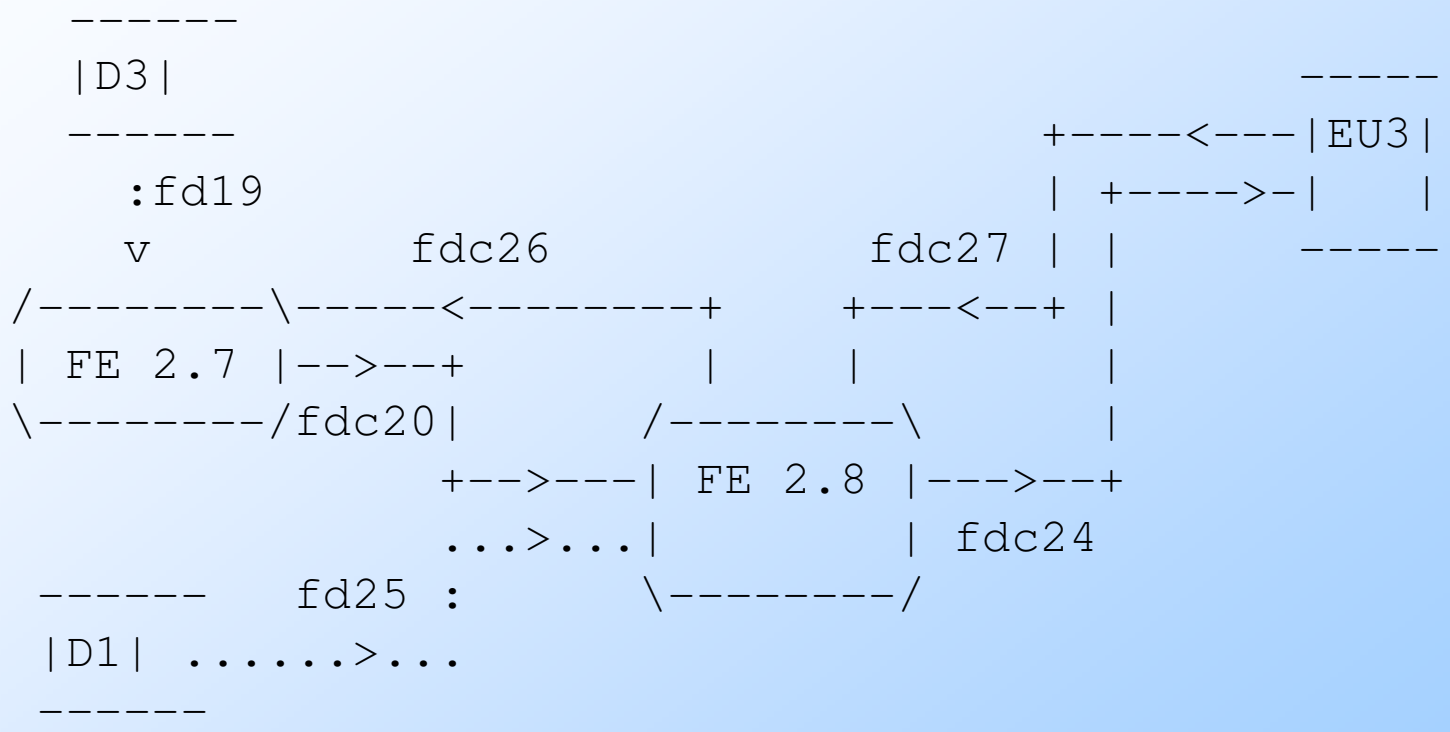
Interpretarea 2

Interpretarea nr. 2: Clientul (EU3) cere raportul de lucru. Pentru aceasta el activeaza functia FE 2.8 care, pentru a putea tipari raportul, are nevoie de datele pe care i le furnizeaza functia FE 2.7. Deci FE 2.8 va activa FE 2.7 care livreaza datele (materiale consumate si detalii lucrare - manopera, etc) dupa care tipareste raportul de lucru.

- **Regula 1:** Multimea de start: { EU3 }
- **Regula 2:** Multimea de stop: { EU3 }
- **Regula 3:** In acest caz FE2.7 produce doar fd20 iar FE2.8 doar fd24. Celelalte fluxuri care ies din ele nu sunt necesare interpretarii. Rezulta ca regula 3 nu are obiect de aplicare in acest caz.
- **Regula 4:** FDC vor fi doar fd20 si fd24.
- **Regula 5:** Se introduce un FDC de la EU3 la FE2.8 ("Activeaza FE2.8-actualizari si rapoarte") si un FDC de la FE2.8 fa FE2.7 ("Activeaza FE2.7 - Acceptare consumuri") deoarece fluxul controlului este in aceasta interpretare invers celui de date, dupa cum se vede si din enuntul interpretarii.

I2: Rezultat

Am simbolizat cu linie punctata FD si cu linie continua FDC. Cele doua FDC adaugate cf. regulii 5 au fost notate fdc26 si fdc27.



Deoarece interpretarea a 3-a nu a fost considerata relevanta, ea nu se ia in consideratie mai departe.

Aplicare R6-R9

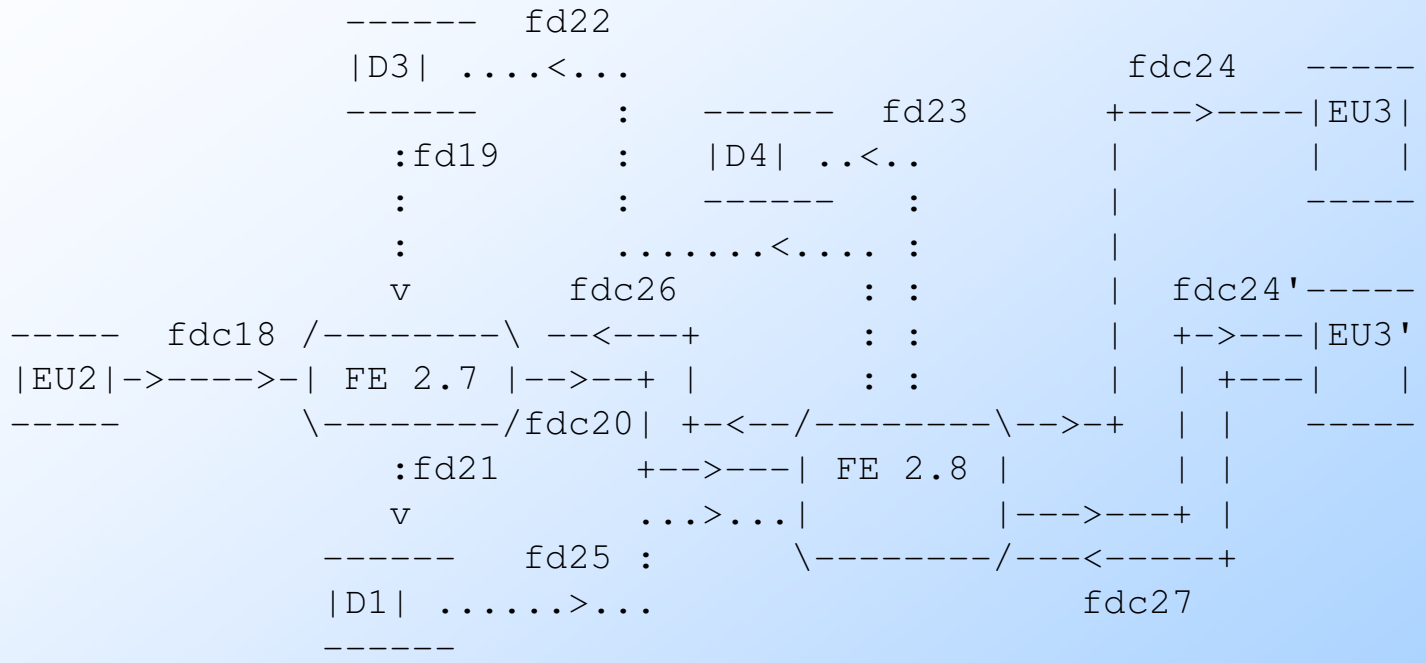
- **Regula 6.** EU3 din interpretarea 2 este in multimea de stop si are un FDC care iese. Rezulta ca vom crea o copie a sa care are nici un FDC care iese, dar are aceleasi fluxuri care intra ca si originalul (unul singur, fdc24).
- **Regula 7.** Vom fuziona diagramele. Diagrama obtinuta contine cite o copie din fiecare flux, functie si date memorate. Fluxurile care sunt FDC intr-una din interpretari vor fi trecute ca FDC, restul raman FD. Diagrama mai contine o copie din EU2 (din interpretarea 1) si trei copii ale EU3 (una din interpretarea 1 si doua din interpretarea 3 + regula 6).
- **Regula 8.** Nu este cazul (Nu avem nici un OR)

Aplicare R6-R9

- **Regula 9.** Avem doua copii ale lui EU3 care au aceleasi fluxuri care ies (in speta nici unul), una din interpretarea 1 si cealalta creata prin regula 6. Ele vor fuziona si hotarim ca va pastra numele initial (EU3). Cealalta copie a lui EU3 (avand un flux care iese) o vom redenumi EU3'.
- Rezultatul obtinut din aplicarea globala a regulilor 1-9 este in slidul urmator, si avem:

1. Multimea de start = { EU2, EU3' }
2. Multimea de stop = { EU3 }

Rezultat final



Partea a 2-a a metodologiei – in cursul urmator.

Proiectarea tranzactiilor AF.

- In cele ce urmeaza, *tranzactiile AF* vor fi numite pe scurt *tranzactii.*
- Acest capitol cuprinde descrierea procesului de proiectare a unei tranzactii. Sunt definite auomatele finite (AF) care vor fi folosite in continuare ca primitive in implementarea tranzactiilor.
- Deoarece AF nu pot modela executia concurenta (sunt secventiale), s-a introdus restrictia ca o tranzactie nu poate fi declansata concurent de catre doua entitati externe.
- Totusi, existenta functiilor logice AND intre FDC in descrierea unei tranzactii duce la o concurenta conceptuala in interiorul sau.

Proiectarea tranzactiilor AF.

- Prezentarea proiectarii unei tranzactii se va face in doi pasi:

1. Se prezinta la inceput rezolvarea problemei pentru tranzactii fara AND, numite si tranzactii secventiale, pentru care se arata cum se defineste un AF asociat
2. Se trateaza apoi cazul tranzactiilor care contin AND. Acestea se vor inlocui cu o multime de subtranzactii secventiale, interconectate prin "functii administrative". Pentru fiecare subtranzactie se va asocia apoi un AF.

Automate finite (AF)

- AF folosite in acesta prezentare sunt cele clasice (se mai numesc si masini secventiale), avand ***actiuni*** (operatii) asociate tranzitiilor dintr-o stare in alta.
- Vom folosi automate deterministe, avand o stare initiala si o multime de stari finale.
- Operatiile asociate cu tranzitiile fac parte din definitia masinii si sunt initiate de tranzitie si nu de procese externe.
- Un AF nu este comun mai multor procese.
- Singura deosebire fata de modelul clasic este ca o parte a intrarilor sunt generate de masina (deci de operatiile asociate tranzitiilor).

Definitie AF

- Un **automat finit** este un sistem (S, E, A, T, s_0, e_0, F) unde:

1. S este multimea starilor, nevida si finita.
2. E este multimea intrarilor, finita (intrarea vida, notata v, apartine lui E).
3. A este multimea actiunilor, finita. O actiune a $\in$ A:
   - preia intrarea curenta si date din afara AF
   - le proceseaza
   - produce date catre exteriorul AF si intrarea urmatoare a masinii. Este posibil ca intrarea urmatoare sa fie ν (intrarea vida).
4. T este functia de tranzitie,

   $$T : S \times E \rightarrow S \times A$$

   Daca T(s, e) = (s', a) atunci fiind in starea s si citind intrarea e, automatul trece in starea s' si executa actiunea a. Daca e = ν masina se opreste.
5. $s_0 \in$ S este starea initiala
6. $e_0 \in$ E este intrarea initiala
7. F este multimea de stari finale, F $\subseteq$ S.

Acceptare

- Se spune ca un AF ***accepta*** secventa de intrare $e_0, e_1, \ldots, e_n$ daca si numai daca:
- - n este finit
- - AF aflat initial in starea s_0 si primind intrarea e_0, face o succesiune de tranzitii (care genereaza intrarile $e_1, \ldots, e_n$) si, in momentul citirii intrarii e_n, se opreste intr-o stare $s \in F$.

Implementarea tranzactiilor secventiale

- O tranzactie secventiala este o tranzactie AF in care FDC nu sunt conectate prin vreun AND.
- AF asociat cu astfel de tranzactii se defineste in felul urmator :

1. Multimea de stari S:

- a. Pentru fiecare obiect O_i care nu apartine multimii de start se asociaza o stare s_i.
- b. Starea s_0 (initiala) se asociaza tuturor elementelor din multimea de start.

E, A

2. Multimea de intrari, E:

a. Pentru fiecare flux de date si control, FDC_i, se asociaza o intrare e_i.

b. Intrarea e_0 se asociaza lui λ (multimea pseudo-FDC descrisa anterior)

3. Multimea de actiuni A:

a. Executia fiecarui obiect o_i (definita anterior) are asociata o actiune a_i

b. Se adauga la aceasta o actiune a_0: "determina care entitate din multimea de start este activata si executa actiunea asociata acesteia" (pentru cazul in care multimea de start contine mai multe elemente).

T

4. Functia de tranzitie, T:

a. Daca exista un FDC de la Oi la Oj, atunci $T(s_i, e_i) = (s_j, a_j)$ unde

- s_i este starea asociata lui o_i
- s_j este starea asociata lui o_j
- e_i este intrarea asociata FDC care leaga o_i de o_j
- a_j este executia lui o_j

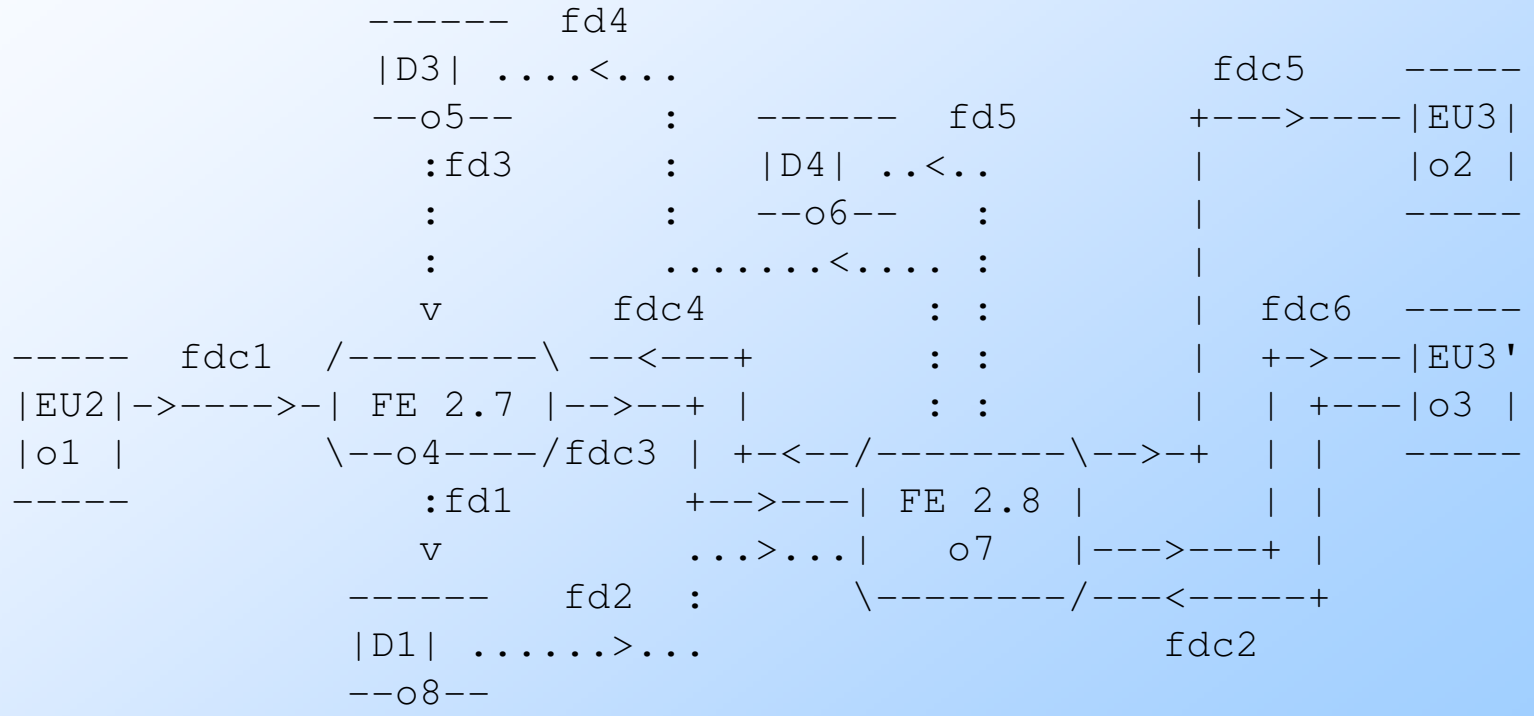
b. $T(s_0, e_0) = (s_0, a_0)$, cu a_0 ca mai sus.

s_0, e_0, F

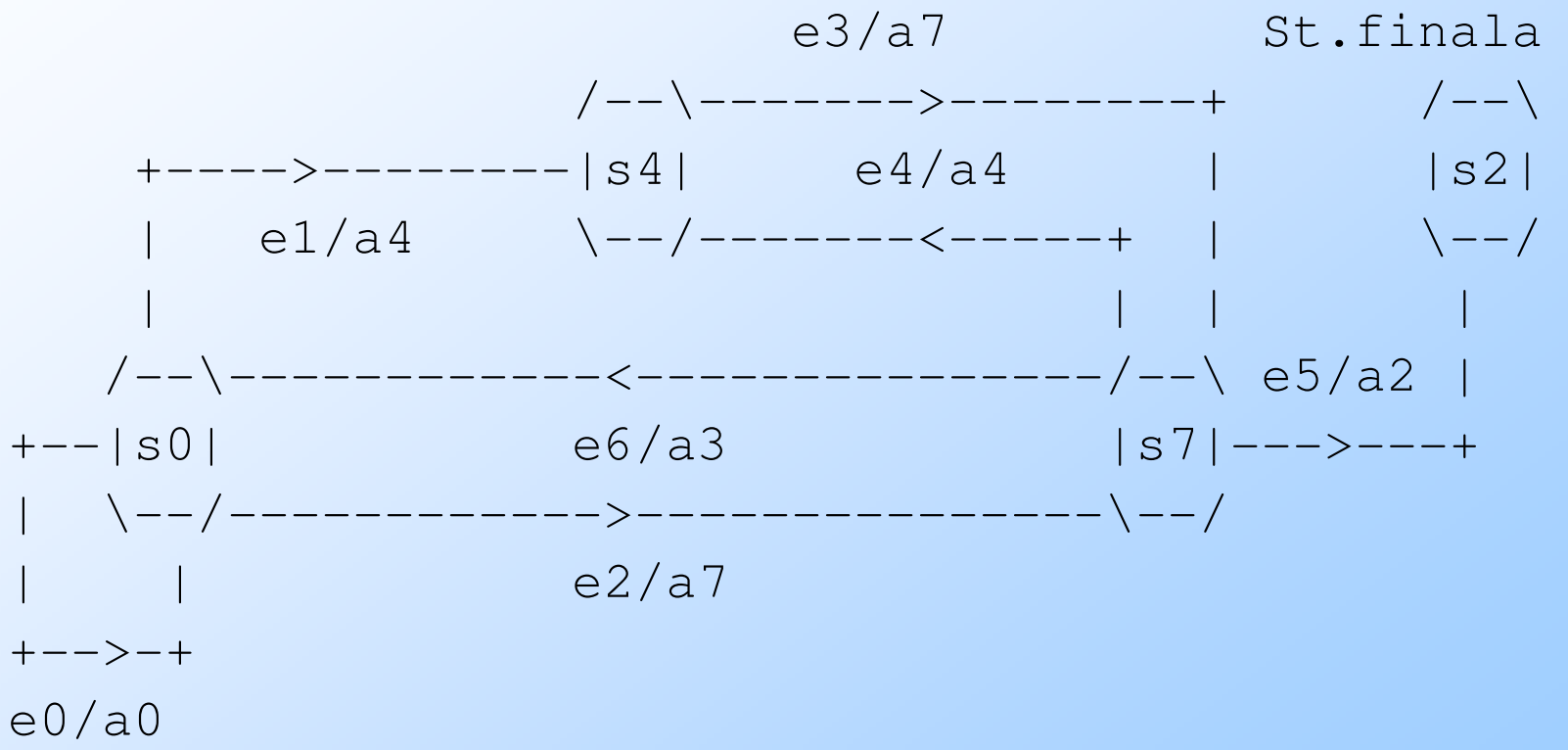
5. **Starea initiala** este s_0, definita mai sus la 1.
6. **Intrarea initiala** este e_0, definita mai sus la 2.
7. **Multimea de stari finale**, F, contine starile asociate obiectelor din multimea de stop a tranzactiei.

Exemplu

Pentru exemplul anterior numerotam obiectele:



Rezultat



Rezultat

- s_0 = starea asociata lui o_1 si o_3
- e_0 = asociat lui λ (eveniment extern, preudo FDC)
- a_0 = determina care dintre { o_1, o_3 } este declansatorul si executa actiunea asociata acestuia.
- e_1/a_4 = corespunzator lui fdc_1, intre o_1 si o_4; se executa a_4, actiunea asociata lui o_4 (functia elementara FE2.7).
- Analog: e_2/a_7, e_3/a_7, e_4/a_4, e_5/a_2, e_6/a_3 corespund FDC cu numerele 2, 3, 4, 5 si 6, executandu-se actiunile: a_4 (fct.2.7), a_7 (fct.2.8), a_2 (stop) si a_3 (entitatea EU3' - emitere semnal vid).
- Observatie: s-au eliminat starile imposibil de atins (corespunzatoare datelor memorate nelegate prin FDC de restul diagramei).

Implementarea tranzactiilor continand AND

- In acest caz, pasii care trebuie urmati sunt urmatorii:

1. Definirea grafului asociat tranzactiei
2. Descompunerea acestuia intr-o multime de subgrafuri de tranzactii secventiale
3. Asocierea de tranzactii secventiale fiecarui subgraf
4. Asocierea unui AF cu fiecare tranzactie secventiala astfel rezultata.

Obiecte administrative

- In subtranzactiile obtinute la punctul 3 sunt introduse urmatoarele 4 tipuri de "obiecte administrative":
- ***a. Entitate de intrare***: Cand punctul de intrare al subgrafului provine dintr-un AND se introduce o astfel de entitate care face transmiterea fluxului de control de la tranzactia apelanta la cea apelata.
- ***b. Functie de transmisie:*** Transmite mai departe fluxul primit. Ea se foloseste in general pentru a evita conectarea directa a doua entitati.

Obiecte administrative

- ***c. Functie de apelare*** (calling function): Executa practic un AND, astfel:
  - Activeaza tranzactiile implicate in acel AND (cele obtinute prin eliminarea AND-ului si setarea punctelor de intrare la arcele implicate)
  - Asteapta terminarea tuturor tranzactiilor activate si trimite in acel moment un semnal entitatii de iesire corespunzatoare
- ***d. Entitate de iesire***: Specifica sfirsitul tuturor tranzactiilor corespunzatoare unui AND.

- Mai multe amanunte privind procesul de implementare: in documentatia ADISSA (cf. bibliografiei)

Consistenta intre specificatie si proiect

Principalele teoreme implicate in aceasta demonstratie sunt urmatoarele:

A. Tranzactii secventiale.

Teorema 1. Toate executiile unei tranzactii secventiale sunt implementate de AF corespunzator si nimic altceva.

Rezulta deci ca, daca exista o implementare exacta a AF, aceasta teorema demonstreaza ca implementarea fluxului de control al tranzactiei AF de catre AF este consistenta cu specificatia tranzactiei.

Consistenta intre specificatie si proiect

B. Tranzactii continand AND.

Teorema 2. Orice executie a unei tranzactii AF este executia multimii subtranzactiilor produse si nimic altceva.

Rezulta ca algoritmul de descompunere de la 5.3. produce un set de AF consistent cu tranzactia AF initiala.

Consistenta intre specificatie si proiect

C. Implementarea tranzactiilor AF prin AF

Teorema 3. Orice executie a tranzactiei AF este executia multimii de AF produse si nimic altceva.

Rezulta ca daca exista o implementare exacta a AF, teorema demonstreaza ca nu numai proiectul (multimea de subtranzactii AF) ci si implementarea (multimea de AF) este consistenta cu specificatia tranzactiei.

Aceasta ultima teorema este rezultatul direct al teoremelor 1 si 2.

Concluzii

- Toata metodologia descrisa se poate implementa sub forma unui sistem de proiectare asistata a sistemelor informatice (un generator de sisteme informatice interactive).
- Din punct de vedere al performantelor sistemelor informatice care ar rezulta ca urmare a aplicarii acestor metode, ele nu au nici un motiv sa fie mai lente decit cele realizate pe alte cai.
- Codul unui automat finit este extrem de simplu, el neputand fi motiv de intirziere. Singura problema o pun functiile elementare (cele care acceseaza date memorate mai ales), dar acestea implementeaza functiuni ale sistemului cerut de utilizator, deci vor fi prezente in orice implementare a sistemului informatic respectiv.
- Avantajul prezentat de aceasta abordare este flexibilitatea deosebita a metodei, permitand dezvoltarea si intretinerea sistemelor informatice cu costuri mult reduse.

Bibliografie

- Gilbert Babin, François Lustman, Peretz Shoval: *Specification and Design of Transactions in Information Systems: A Formal Approach*, IEEE Transactions on Software Engineering, Volume 17, Issue 8 (August 1991)

Sfârşitul capitolului 11