Dezvoltare cerinte

Adapted after :

Timothy Lethbridge and Robert Laganiere,

Object-Oriented Software Engineering –

Practical Software Development using UML and Java, 2005

(chapter 4)

4.1 Analiza de domeniu

***Analiza de domeniu* este procesul prin care inginerul software invata cunostinte din domeniul aplicatiei cu scopul de a intelege mai bine problema de rezolvat.**

- *Domeniul* este sfera de activitate economica sau tehnologica in care va fi utilizat produsul software.
- Un *expert in domeniu* este o persoana care intelege foarte bine domeniul (de aplicatie)

Beneficiile analizei de domeniu:

- Dezvoltare mai rapida
- Un sistem mai bun
- Anticiparea extensiilor

Structura unui document de analiza a domeniului

A. Introducere
B. Glosar
C. Cunostinte generale despre domeniu
D. Clienti si utilizatori
E. Sistemul si echipamentele utilizate
F. Sarcini si proceduri efectuate
G. Produse software similare existente
H. Similaritati cu alte domenii

4.2 Punctul de pornire al proiectelor software

Patru mari categorii de proiecte software:

	Cerintele trebuie sa fie determinate	Cerintele au fost produse de clienti
Proiect complet nou	A	B
Evolutie a unui sistem existent	C	D

4.3 Definirea problemei

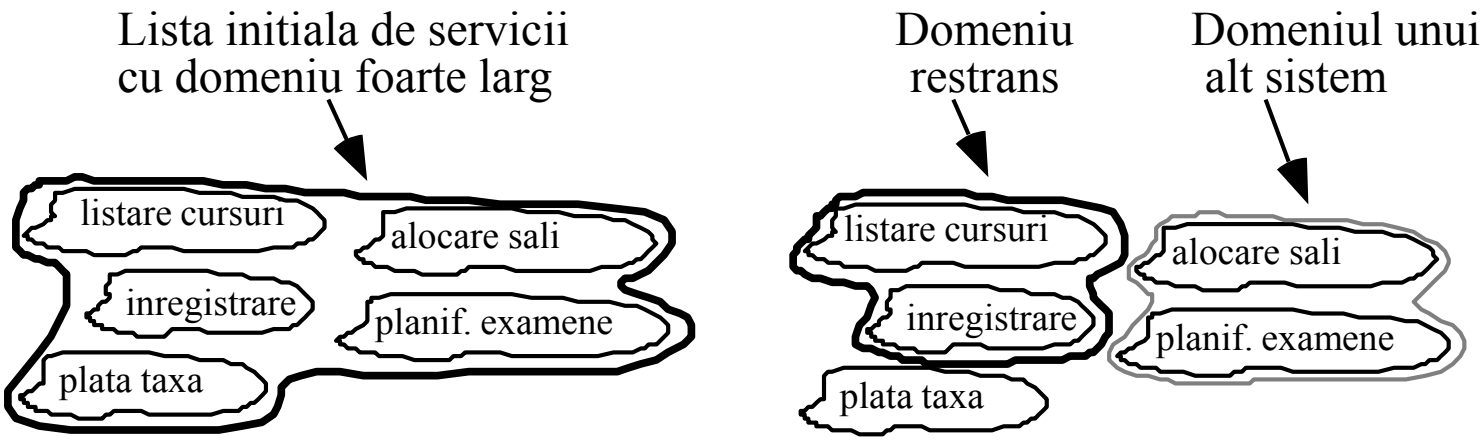
- **Procesul de determinare a cerintelor incepe prin definirea *problemei* de rezolvat.**
- **O problema poate fi exprimata ca:**
  - O *dificultate* intampinata de utilizatori sau clienti,
  - O *oportunitate* de imbunatatire a productivitatii sau a vanzarilor.
- **Solutia problemei implica dezvoltarea unui produs software.**
- **Este bine ca formularea problemei sa fie succinta – preferabil una sau doua fraze.**
  - Exemplu:
    - —Sistemul va permite unui student sa se inregistreze la cursuri si sa isi modifice inregistrarea, simplu si rapid. De asemenea, va asista studentii in alegerea cursurilor pe care le considera a fi cele mai interesante si mai utile.

Limitarea domeniului problemei

Este important ca domeniul problemei sa fie definit cat mai precis

- Se alcatuieste o lista a serviciilor posibile
  - —Unele se exclud daca lista initiala este prea larga
  - —Se identifica scopurile principale (ale clientului / utilizatorului) daca lista initiala este prea ingusta

Exemplu: Sistem de gestiune date universitare



4.4 Ce este o cerinta

- **O *cerinta* este 1) o descriere a unui serviciu pe care sistemul trebuie sa il furnizeze sau 2) o descriere a unei constrangeri pe care sistemul trebuie sa o satisfaca. Implementarea oricarei cerinta trebuie sa contribuie la rezolvarea problemei clientului / utilizatorului; pachetul de cerinte reprezinta rezultatul unui acord intre toti participantii la proiect.**

- **O colectie de cerinte este un *document de cerinte*.**

4.5 Tipuri de cerinte

Cerinte functionale

- Descriu serviciile sistem solicitate de client / utilizator (*ce* ar trebui sa faca sistemul software)

Cerinte nonfunctionale

- Descriu constrangeri sub care trebuie sa opereze sistemul sau standarde ce trebuie sa fie satisfacute de sistemul software

Cerinte functionale

Cerintele functionale descriu *servicii* furnizate catre *utilizator* sau catre *alte sisteme*.

Cerintele functionale pot sa descrie:

- Ce *intrari* trebuie sa accepte sistemul
- Ce *iesiri* trebuie sa produca sistemul
- Ce date (care pot fi utilizate de alte sisteme) trebuie sa fie *stocate* de sistem
- Ce *calcule* trebuie sa efectueze sistemul
- *Temporizarea sau sicronizarea* serviciilor (in special in cazul sistermelor de timp real)

Cerinte nonfunctionale

Trei mari categorii

1. Cerinte care reflecta *atribute de calitate*:
   - —Timp de raspuns
   - —Utilizare resurse
   - —Fiabilitate
   - —Recuperare din erori
   - —Mentenabilitate
   - —Reutilizabilitate
2. Cerinte legate de platforma si tehnologia de calcul
3. Cerinte legate de metodologie (procesul de dezvoltare), costuri si date de livrare

4.6 Cazuri de utilizare

- **Un *caz de utilizare* (engl. *use case*) este o descriere a unui set de secvente de actiuni pe care le efectueaza un sistem pentru a produce un rezultat observabil si semnificativ pentru un actor particular [BRJ99].**
  - Un *model de cazuri de utilizare* consta din
    - — O colectie de cazuri de utilizare
    - — O colectie de actori (un *actor* reprezinta un tip de utilizator (nu neaparat uman) care interactioneaza cu sistemul; in interactiunea cu sistemul, fiecare actor joaca un set coerent de roluri [BRJ99,JBR99])
    - — O descriere (diagramatica) a modului in care aceste componente interactioneaza
  - Pentru ca un model de cazuri de utilizare sa fie inteligibil este necesara gruparea secventelor de actiuni similare intr-un singur caz de utilizare.

Cazuri de utilizare

- In general, un caz de utilizare acopera intreaga secventa de actiuni asociata unei sarcini de calcul (serviciu furnizat de sistem).
- Un caz de utilizare descrie *interactiunea utilizatorului* cu sistemul...
  —nu calculele efectuate de sistem.
- Descrierea cazurilor de utilizare trebuie sa fie cat se poate de independenta de designul interfatei utilizator.

Scenarii

- **Un scenariu este o *instanta* a unui caz de utilizare, care implica**
  - —o anumita instanta a unui actor
  - —un anumit (moment / o anumita durata de) timp
  - —date specifice.

Cum se descrie un caz de utilizare

A. Nume: se alege un nume scurt, descriptiv

B. Actori: Se listeaza actorii care utilizeaza cazul de utilizare.

C. Scopuri: Ce urmaresc actorii care utilizeaza cazul de utilizare.

D. Preconditii: Starea sistemului inainte de efectuarea cazului de utilizare.

E. Descriere: Scurta descriere informala a cazului de utilizare.

F. Relatii cu alte cazuri de utilizare

G. Actiuni / pasi: Se descrie fiecare actiune / pas (pe 2 coloane: actiuni actor, raspunsuri sistem).

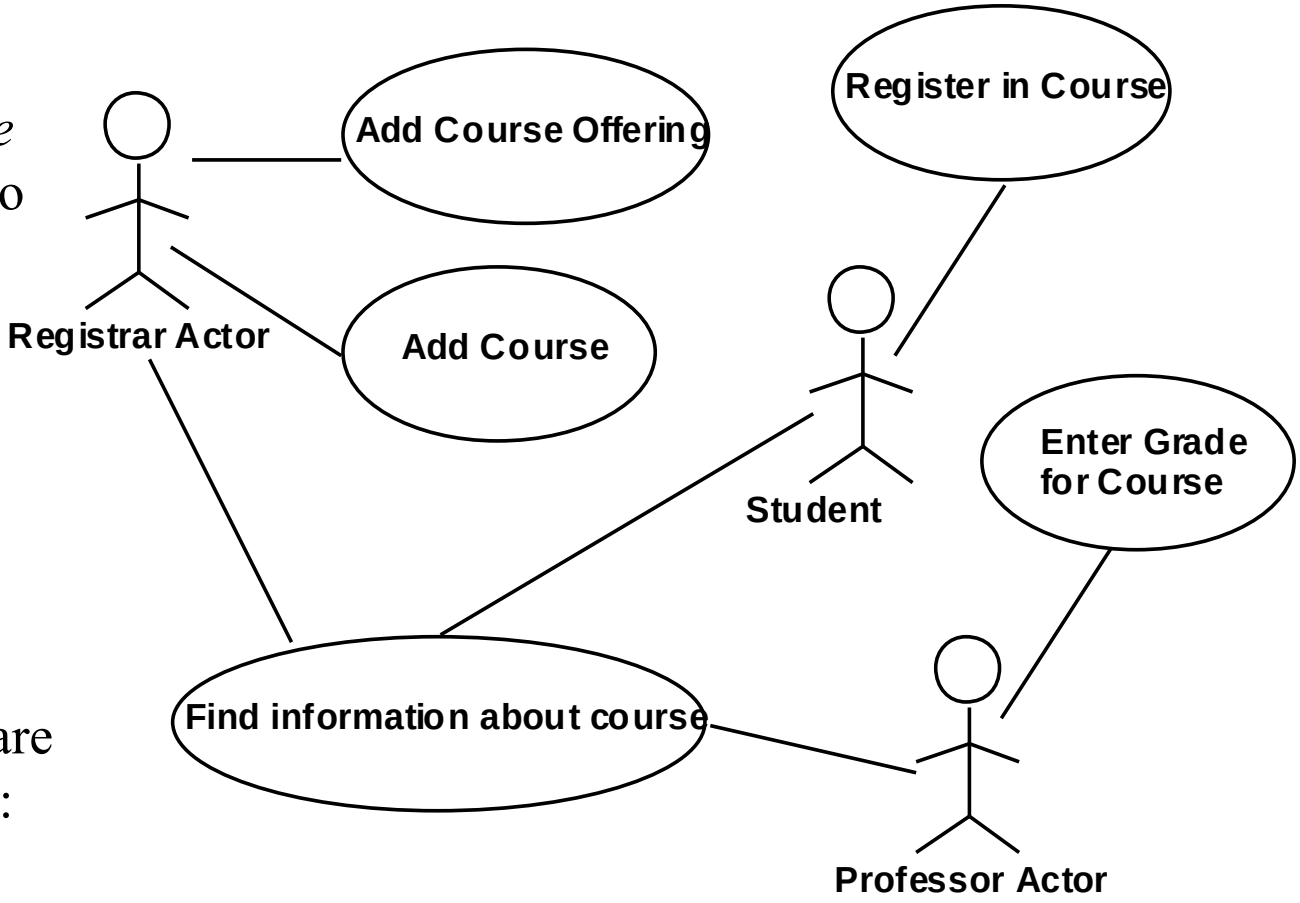
H. Postconditii: Starea sistemului dupa terminarea cazului de utilizare.

• Numai numele (A) si actiunile atasate (G) sunt esentiale.

—Intr-o descriere simplificata se pot omite celelalte componente.

Un exemplu de diagrama de cazuri de utilizare

Grafic
• un *caz de utilizare* este reprezentat ca o elipsa,
• un *actor* este reprezentat sub forma unui omulet (din bete).



Diagramele de cazuri de utilizare contin [BRJ99]:
• actori,
• cazuri de utilizare,
• asocieri, generalizari si dependente.

Extensii

- Utilizate pentru explicitarea interactiunilor *optionale* sau pentru tratarea cazurilor de *exceptie*.
  - —Prin crearea de cazuri de utilizare care reprezinta extensii descrierea cazului de baza ramane simpla.
  - —In extensie trebuie
    - fie sa se listeze toate actiunile (toti pasii) de la inceputul cazului de utilizare pana la sfarsit (inclusiv situatia optionala sau de exceptie),
    - fie sa se indice care este punctul de extensie (punctul in care extensia modifica secventa de baza).

Generalizari

- *Generalizarile* (/ *specializarile*) se comporta la fel ca in diagramele de clase (si utilizeaza aceeasi notatie grafica).
  - —Un caz de utilizare mai general poate reprezenta cateva cazuri de utilizare similare.
  - —Unul sau mai multe cazuri de utilizare pot oferi specializari ale comportamentului general.

Incluziuni

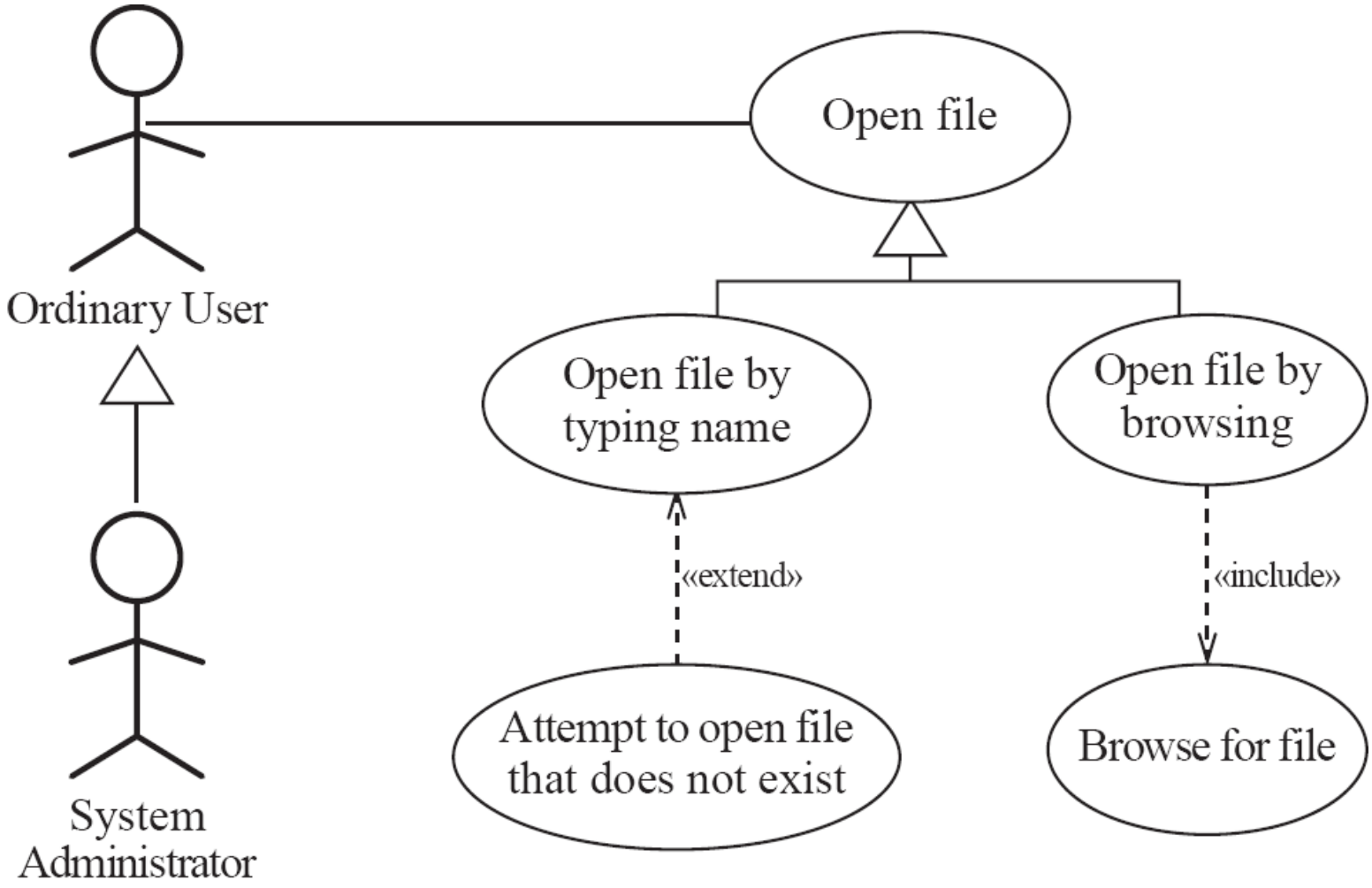
- Permit reprezentarea unui *comportament comun* pentru mai multe cazuri de utilizare.

  —Chiar si cazuri de utilizare foarte diferite pot partaja secvente de actiuni.

  —Permit evitarea repetarii unor detalii in mai multe cazuri de utilizare.

  —Reprezinta sarcini de calcul secundare (de nivel mai coborat).

Exemplu de diagrama de cazuri de utilizare cu generalizare, extensie si incluziune



Exemplu de descriere

Caz de utilizare: Open File

Relatii:
Generalizare pentru:
• Open file by typing name
• Open file by browsing

Actiuni:

Actiuni actor:	**Raspunsuri sistem:**
1. Alegere comanda ‘Open…’	2. Afisare caseta dialog ‘File open’
3. Specificare nume fisier	
4. Confirmare selectie	5. Eliminare caseta dialog

Exemplu (continuare)

Caz de utilizare: Open file by typing name

Relatii:
Specializare pentru:
- Open file

Actiuni:

Actiuni actor	**Raspunsuri sistem**
1. Alegere comanda ‘Open…’	2. Afisare caseta dialog ‘File open’
3a. Selectare camp text	
3b. Introducere nume fisier	
4. Selectare ‘Open’	5. Eliminare caseta dialog

Exemplu (continuare)

Caz de utilizare: Open file by browsing

Relatii:
Specializare pentru: Open file
Include: Browse for file

Actiuni:

Actiuni actor:	**Raspunsuri sistem:**
1. Alege comanda ‘Open…’	2. Afisare caseta dialog ‘File open’
3. Browse for file (caz de utilizare inclus)	
4. Confirmare selectie	5. Eliminare caseta dialog

Exemplu (continuare)

Caz de utilizare: Attempt to open file that does not exist

Relatii:
Extensie pentru: Open file by typing name

Actiuni:

Actiuni actor:	**Raspunsuri sistem:**
1. Alege comanda ‘Open…’	2. Afisare caseta dialog ‘File open’
3a. Selectare camp text	
3b. Introducere nume fisier	
4. Selectare ‘Open’	5. Se indica faptul ca fisierul nu exista
6. Corectare nume fisier	
7. Selectare ‘Open’	8. Eliminare caseta dialog

Exemplu (continuare)

Caz de utilizare: Browse for file (incluziune)

Actiuni:

Actiuni actor:	**Raspunsuri sistem:**
1. Daca fisierul dorit nu este afisat se selecteaza un director	2. Afisare continut director
3. Se repeta pasul 1 pana cand se afiseaza fisierul dorit	
4. Selectare fisier	

Cazuri de utilizare prioritare

- Adesea unul sau mai multe cazuri de utilizare pot fi selectate ca fiind principale (esentiale) pentru sistem
  - De exemplu, intr-un sistem de rezervari pentru curse aeriene cazul de utilizare principal va fi ‘Rezervare loc pentru un anumit zbor’
    - » Intregul sistem poate fi construit in jurul acestui caz de utilizare.
- Exista si alte motive pentru tratarea cu prioritate a anumitor cazuri de utilizare:
  - —Implementarea anumitor cazuri de utilizare poate implica un inalt grad de *risc* (tehnic sau managerial)
  - —Anumite cazuri de utilizare pot avea o mare *valoare comerciala sau politica*

Beneficii ale dezvoltarii ghidate de cazurile de utilizare

- Cazurile de utilizare ajuta in definirea *domeniului* sistemului.
- Cazurile de utilizare pot fi utilizate ca instrument de *planificare* a procesului de dezvoltare.
- Cazurile de utilizare permit atat dezvoltarea cat si validarea cerintelor.
- Pe baza cazurilor de utilizare se pot proiecta cazuri de test.
  - —Un caz de test este o specificare a unui scenariu de testare a sistemului. Un caz de test specifica ce se testeaza, in ce conditii, cu ce intrari si care sunt rezultatele asteptate [JBR99].
- Cazurile de utilizare pot fi utilizate pentru structurarea manualelor utilizator.

Beneficii ale dezvoltarii ghidate de cazurile de utilizare

- Strategiile de baza pentru testarea de integrare (top-down si bottom-up) sunt aplicabile daca modulele software alcatuiesc o structura ierarhica.
  - —Intr-un sistem OO modulele de baza ce urmeaza a fi testate sunt clasele de obiecte.
  - —Problema este ca, in general, un sistem OO nu este structurat sub forma unei ierarhii de module sau subsisteme [Som01].
- **Testarea bazata pe scenarii sau cazuri de utilizare este adesea cea mai buna strategie pentru testarea de integrare a sistemelor OO [JBR99,Som01].**
  - —In aceasta abordare inginerul software identifica scenarii de testare pe baza cazurilor de utilizare.
    - In general, fiecare caz de utilizare da nastere unui set de cazuri de testare.

4.7 Tehnici de colectare a cerintelor

Observare

- Lecturare documente si discutarea cerintelor cu utilizatorii
- Asistarea si urmarirea utilizatorilor potentiali in timpul lucrului
- Inregistrarea sesiunilor de lucru pe benzi video

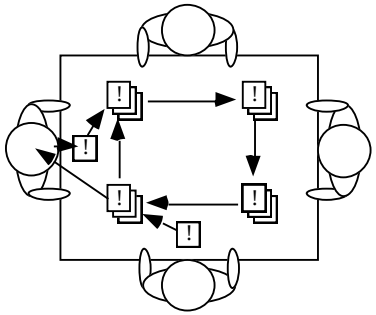
Intervievare

- Organizare interviuri cu intrebari / discutii despre
  - —detalii specifice
  - —viziunea participantilor asupra viitorului sistem
  - —idei sau solutii alternative
  - —alte surse de informatie
  - —reprezentari diagramatice

Colectarea cerintelor...

Brainstorming (sesiuni ce favorizeaza idei si solutii inspirate)

- Se numeste un moderator experimentat
- Participantii sunt dispusi in jurul unei mese
- Se alege o intrebare suport, de exemplu:
  - Ce aspecte sunt importante pentru sistem?
  - Ce surse de date pot fi anticipate?
  - Ce 'iesiri' ar trebui sa produca sistemul?
  - Ce clase sunt necesare pentru reprezentarea domeniului?
- Se cere fiecarui participant sa scrie un raspuns si sa il inmaneze vecinului.



JAD *(Joint Application Development)* este o tehnica bazata pe sesiuni intense de brainstorming

Colectarea cerintelor...

Prototipizare

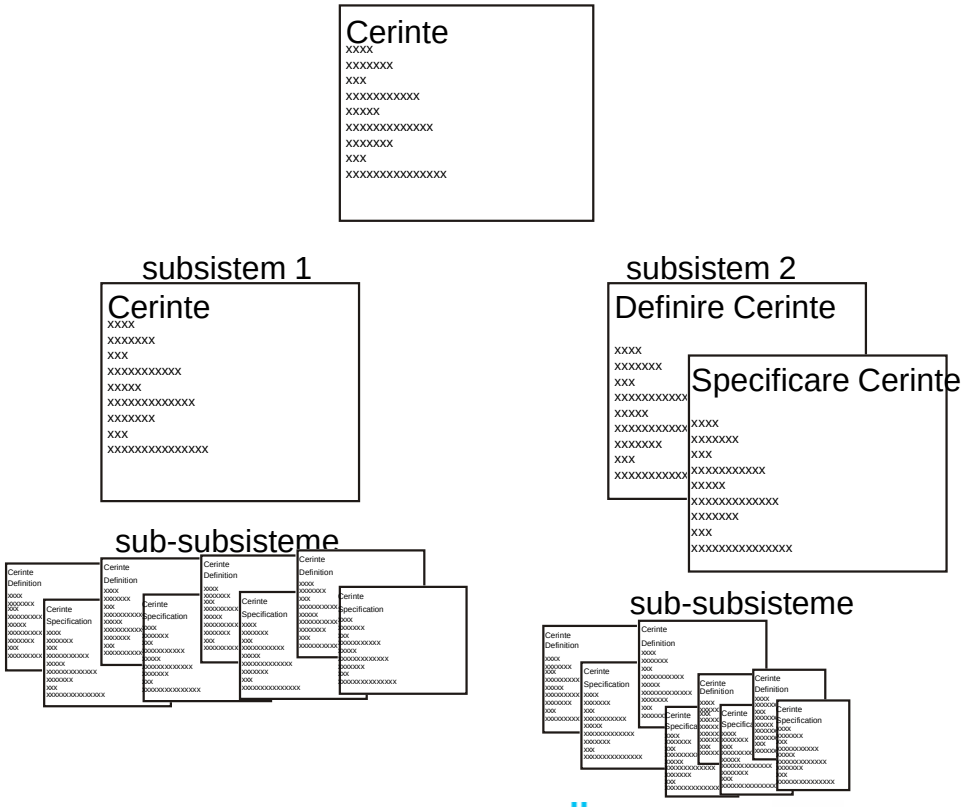
- Cea mai simpla varianta: prototip pe hartie a interfetei utilizator (IU)
  - —Reprezentari grafice sugestive aratate si explicate utilizatorilor
- Cea mai uzuala abordare: un sistem ce simuleaza IU
  - —Scris intr-un limbaj de prototipizare rapida
  - —In mod normal nu efectueaza calcule si nu interactioneaza cu baze de date sau cu alte sisteme
  - —Se pot prototipiza anumite aspecte ale sistemului (algoritmi specifici, operatii pe baze de date, etc.)

4.8 Tipuri de documente de cerinte

Extreme ce ar trebui sa fie evitate:
• O schita informala a cerintelor alcatuita din cateva paragrafe sau diagrame simple.
• O lista lunga de specificatii ce contine mii de pagini cu detalii complicate.

•In mod normal se utilizeaza urmatoarea terminologie:
• *Definire cerinte*
  • un document mai putin detaliat (de nivel inalt).
• *Specificare cerinte*
  • un document mai precis si mai detaliat.

• Documentele de specificare a cerintelor pentru sistemele de dimensiuni mari sunt in mod normal structurate ierarhic



Nivelul de detaliere in documentele de cerinte

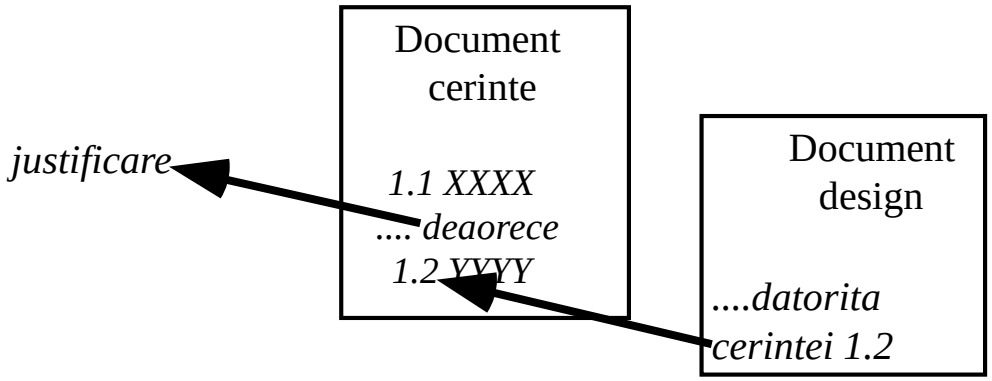
- Nivelul de detaliere in documentele de cerinte depinde de:
  - —Dimensiunea sistemului
  - —Necesitatea de interfatare cu alte sisteme
  - —Stadiul in colectarea cerintelor
  - —Experienta in domeniul de aplicatie si in utilizarea tehnologiei
  - —Costurile implicate de cerinte eronate

4.9 Recenzia documentelor de cerinte

- **Fiecare cerinta individuala trebuie sa fie verificata cu atentie. Fiecare cerinta individual ar trebui:**
  - —Sa conduca la beneficii care sa depaseasca costurile de dezvoltare
  - —Sa fie importanta pentru solutionarea problemei
  - —Sa fie exprimata intr-o notatie clara si consistenta
  - —Sa nu fie ambigua
  - —Sa fie logic consistenta
  - —Sa conduca la un sistem de calitate
  - —Sa fie realista in conditiile resurselor disponibile
  - —Sa fie verificabila
  - —Sa fie identificabila in mod unic
  - —Sa nu impuna constrangeri prea mari asupra procesului de proiectare

Documente de cerinte…

- Documentele de cerinte ar trebui sa fie:
  - —Suficient de complete
  - —Bine structurate
  - —Clare
  - —Acceptate de toti participantii (clienti si dezvoltatori)
- Intr-un document de design (proiectare) ar trebui sa fie intotdeauna posibil sa se precizeze care cerinta este implementata de fiecare aspect al designului (in limba engleza se utilizeaza termenul *traceability*).



Structura unui document de cerinte

A. Problema

- O descriere succinta a problemei de rezolvat

B. Domeniul

- Referinte la documente de analiza de domeniu, standarde, sisteme similare

C. Context si modele sistem

- Contextul in care opereaza sistemul, descriere globala subsisteme, suport hardware, etc.

D. Cerinte functionale

E. Cerinte nonfunctionale

Referinte suplimentare

[BRJ99] G. Booch, J. Rumbaugh and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[JBR99] I. Jacobson, G. Booch and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[Som01] I. Sommerville. *Software Engineering* (6th edition). Addison-Wesley, 2001.