Arhitectura sistemelor software

Cursul Vi 8-10 săptămânal

Seminarul Vi 10-11 săptămânal (Grupa 1)

2 ore o dată la 2 săptămâni (Grupa 2 în impar / Grupa 3 în par)

Echipa de Teams

Formular / Materiale de curs / Ass. anunțuri pentru proiect(e)

Introducere

Partea I: Analiza arhitecturală

Partea a II-a: Proiectarea arhitecturală

Similaritate Semestrul trecut Semestrul curent

Scopul principal al cursului este acela de a defini o legătură cuantificabilă între analiza și proiectarea arhitecturii aplicației, adică între nevoile arhitecturale ale aplicației și soluțiile arhitecturale propuse

argumente (de ce?)

Proiect (documentație): 50%

Descriere arhitecturală a unei aplicații dezvoltate anterior

Examen scris (în sesiune): 50% 2 ore

întrebări de pentru o aplicație la prima vedere

Arhitectura sistemelor software

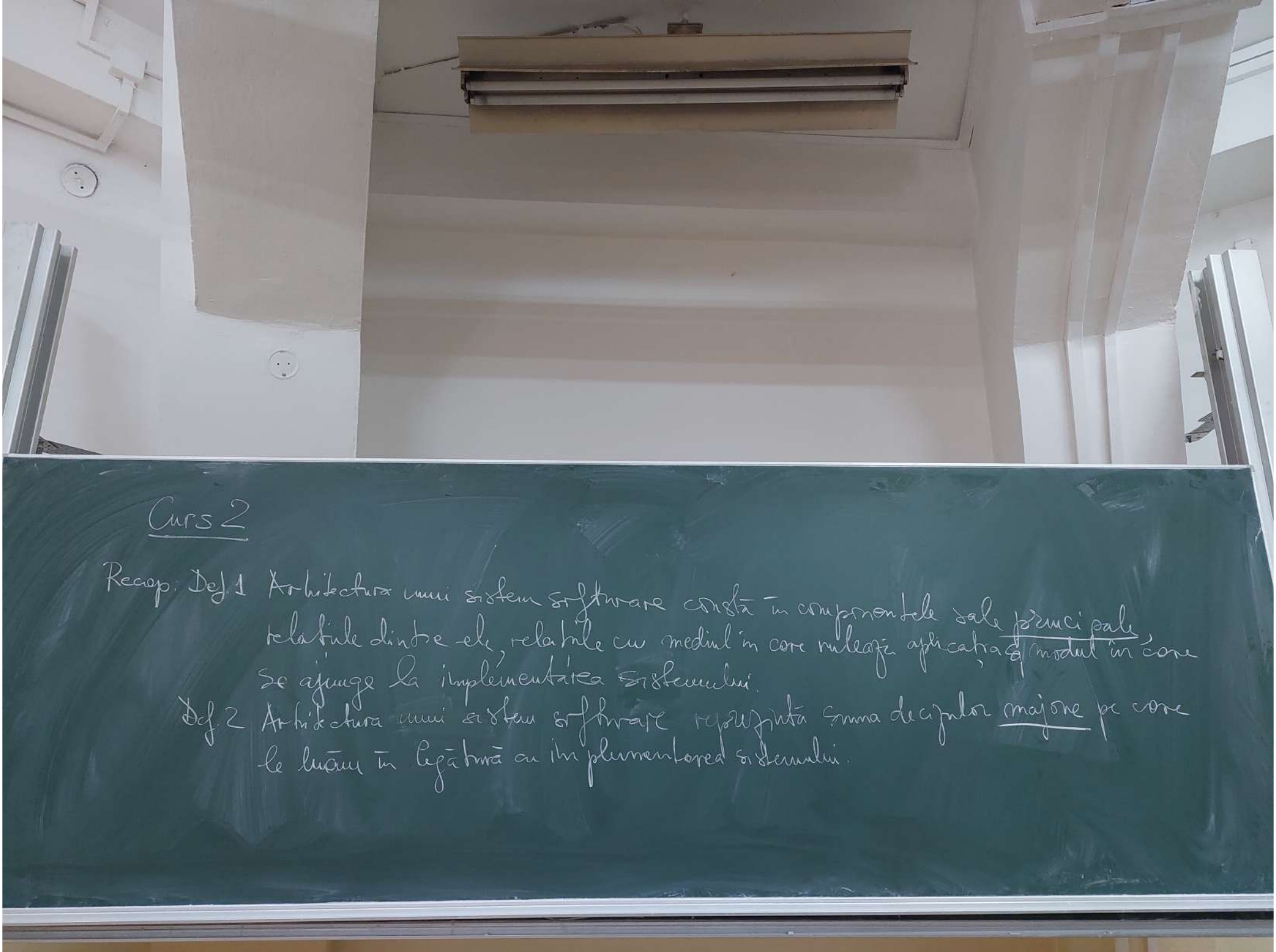
- relațiile dintre componentele principale ale sistemului (ce fel de?)
  - depind de complexitatea aplicației
- relațiile sistemului cu mediul în care va fi utilizat
- modul în care ajungem la implementarea aplicației pe baza arhitecturii propuse (și validitatea arhitecturii)

Proiectarea
- arhitecturală (la nivel centralizat) – arhitect, software
- componentelor (la nivel de programator)

Analiza
- arhitecturală (la nivel centralizat) – analist tehnic
- componentelor (la nivel de analist/programator)
- funcțională (de business)
- tehnică (calitățile aplicației)

Arhitectura unui sistem software reprezintă suma deciziilor strategice și tactice importante luate pentru dezvoltarea sistemului

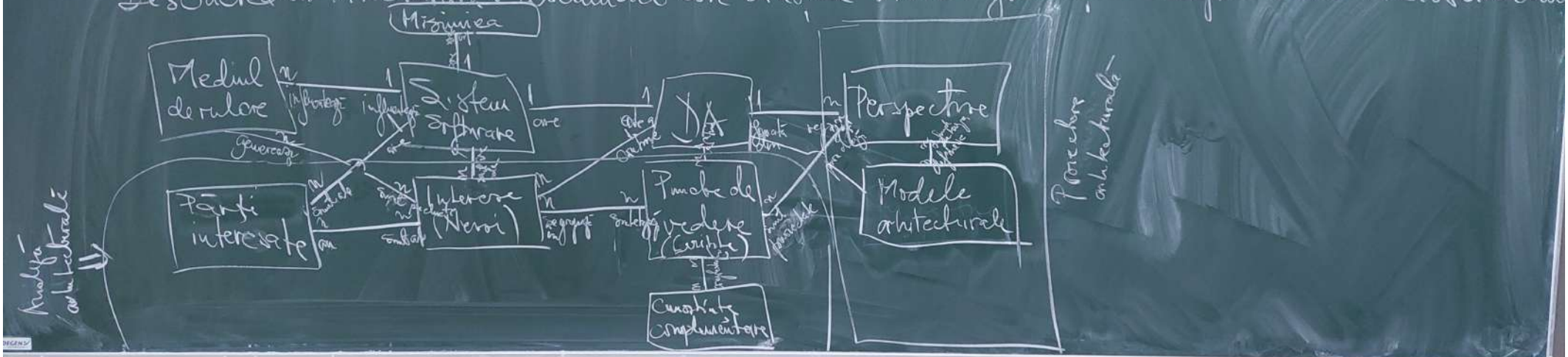
argumentate


Curs 2
Recap. Def. 1 Arhitectura unui sistem software constă în componentele sale principale, relațiile dintre ele, relațiile cu mediul în care rulează aplicația și modul în care se ajunge la implementarea sistemului.
Def. 2 Arhitectura unui sistem software reprezintă suma deciziilor majore pe care le luăm în legătură cu implementarea sistemului.

Def. 2 Arhitectura unui sistem software reprezintă suma deciziilor majore pe care le luăm în legătură cu implementarea sistemului.

Artefactele care intră în descrierea unei arhitecturi software (DA)

Descrierea arhitecturală: document care conține toate informațiile despre arhitectura sistemului



Seminarii

Gr. 1 : Vi 10-11 (săptămânal), sala 215

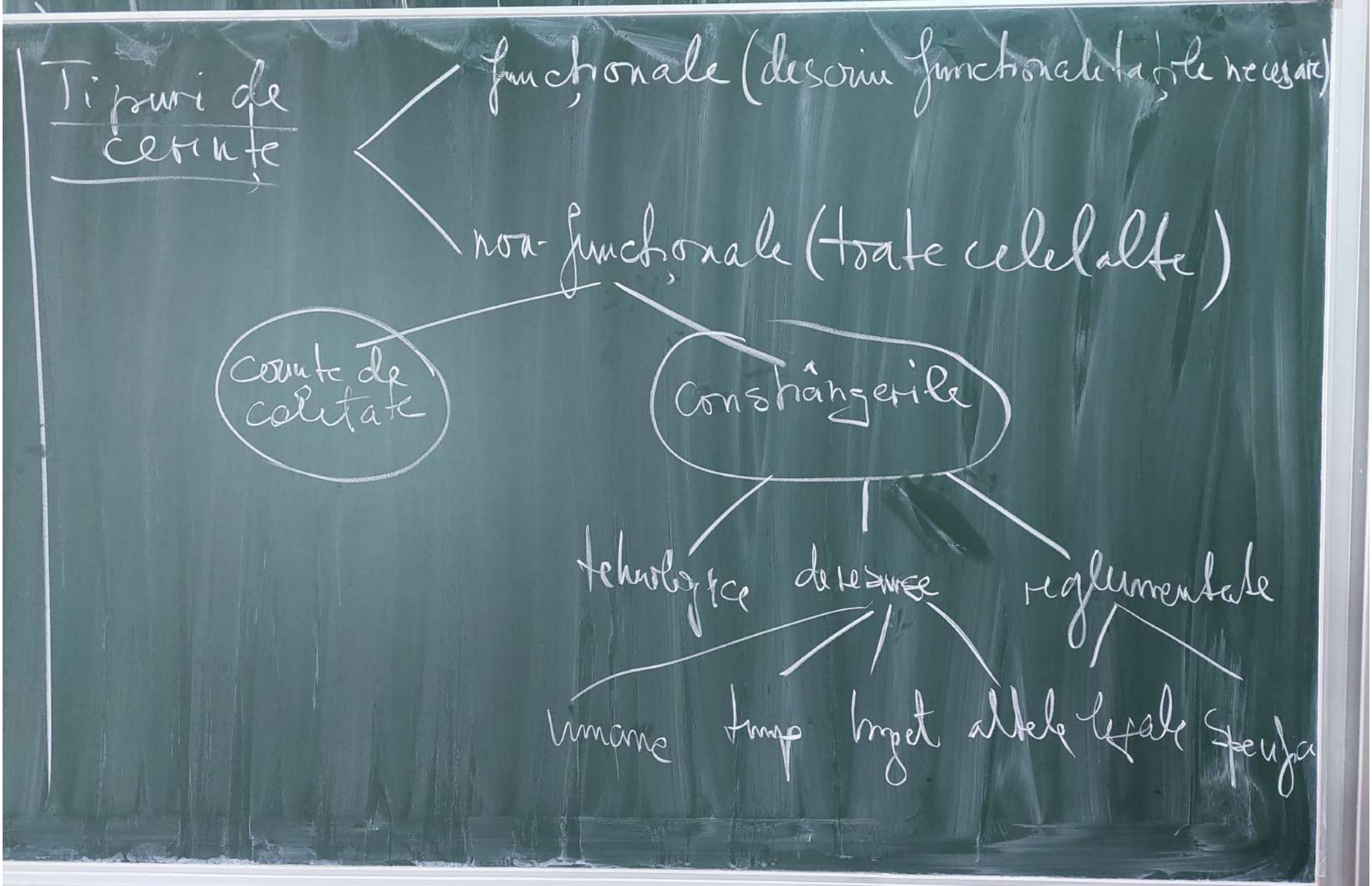
Gr. 2 : Lu 18-20 (săptămână impară), sala 216

Gr. 3 : Pa 18-20 (săptămână impară), sala 216

Recuperare Seminar 1

Gr 2 : Lu 13 martie, 18-20, sala 216

Gr. 3 : Pa 14 martie, 18-20, sala 216


Tipuri de cerințe
funcționale (descriu funcționalitățile necesare)
non-funcționale (toate celelalte)
cerințe de calitate
constrângerile
tehnologice
de resurse
reglementate
umane
timp
buget
alte
legale
specifice

Gr 2: Lu 13 martie, 18-20, sala 216
Gr. 3: Pla 14 martie, 18-20, sala 216

Curs 3 Analiza arhitecturală

Dualitatea Problemă ↔ Soluție
⇑ Analiză ⇑ Proiectare

Colectarea cerințelor (intereselor, nevoilor)

Principiul 1. Nicio aplicație software nu reprezintă un scop în sine. Orice aplicație software reprezintă soluția la o anumită problemă din lumea reală.

Principiul 2: Cerințele aplicației software nu se inventează (nu presupunem că știm ce trebuie să facă) Cerințele se extrag de la persoanele sau grupurile interesate (Stakeholders)

Principiul 3: Cerințele colectate de la părțile interesate se curăță, se completează cu informații specifice colectate din literatura de specialitate, aplicații similare etc și se structurează în Puncte de vedere (Viewpoints)

Principiul 4: Nu toate cerințele au aceeași relevanță pentru dezvoltarea aplicației. Cerințele trebuie prioritizate

Studiu de caz (partial): Motorul de căutare generalist de la Google

Întrebarea: De ce este Google Search Engine una dintre cele mai de succes aplicații software din toate timpurile?

Misiunea aplicației: Oferă ingestii relevante privind locul de unde s-ar obține răspunsul căutat pentru orice subiect de care este interesat orice utilizator

Principalele calități: Simplitatea ⇒ usurința de utilizare
Timpul de răspuns
Relevanța răspunsurilor

Cerințe de calitate (I)

Tipuri de cerințe < funcționale
non-funcționale < de calitate
constrângeri

Motorul de căutare Google – viteza răspunsului
– relevanța răspunsului
– simplitatea interfeței

Performanță

– acoperă orice comportament al aplicației în cazul în care pornește unul sau mai mulți algoritmi ⇒ criteriu de evaluare al implementării acelui algoritm în aplicație
– estimarea condițiilor în care trebuie atins indicatorii de performanță doriți, evoluția acestor condiții în timp
– testarea performanței în condiții „reale"

Percepție și performanță

– cum sunt așteptările utilizatorului?
– tactica de modificare a percepției (diversiune)
– tactica de anticipare a cererii

Disponibilitate

– proprietatea sistemului de a răspunde la defect
indisponibilitate ← nefuncționarea sistemului sau a unei anumite componente cu percepția unei lipse de performanță acute

Strategii de răspuns

– detectia + înregistrare
– notificare
– restart (LOL)

recuperarea
↘ relocarea componentei afectate
↳ nu interesează decât cea ce ține de arhitectura aplicației

– redundanță – pasivă – backup
– activă
– mecanisme (shadowing)

– întreruperile programate!!!

Cerințe de calitate (II)

Securitate

- trebuie adaptată la nevoile reale ale aplicației
- nicio metodă de protecție nu este garantată 100%
- măsurile avansate de securitate generează costuri suplimentare

atacatorii pentru infractori
pierderile neajunsuri ale unui atac reușit
- date sensibile: bancare, personale
- acțiuni sensibile (care interacționează cu ființa umană)

Strategii

- Prevenția (implementată gradual și adecvat) → dificultatea constă în diversitatea enormă a atacurilor posibile
- Detecția / Înregistrare / Notificare
- Recuperare din atac: recuperare stări sistemului din copii salvate anterior; shadowing (sincronizarea stărilor diverselor componente ale sistemului)

Experiența de utilizare (Usability) UX/UI

- atractivitate
- aplicația pune accent pe utilizator
- ușurința de utilizare
- eficiența de utilizare (Efficiency)
- eficacitatea de utilizare (Effectiveness)
- ușurința de învățare

- experiența de utilizare trebuie adaptată grupului țintă (profilarea utilizatorilor)
  - utilizatori experimentați
  - utilizatori neexperimentați
  - cutume de utilizare

Securitate vs ~~ușurința~~ experiența de utilizare

- măsuri de securitate
  - parole complexe
  - reintroducerea periodică a parolei
  - certificate de securitate

Problema principală este generată de interconectarea sistemelor software

Constrângeri

Recap: cerințe de calitate ← Performanță, Disponibilitate, Experiența de utilizare, Securitate

Există și altele (destul de multe)

- → Mentenabilitate ← upgrade-ul mediului de rulare; upgrade la aplicație
- → Modificabilitate – raportul dintre modificări și costuri
  - ↳ riscuri ← costuri – provine din multiplicarea mai multor modificări; alterarea arhitecturii în timp (alterarea indicatorilor de calitate)
- → Testabilitate
- → Scalabilitatea ← în volum (utilizatori, date etc); în complexitate (funcționalități noi)
- → Portabilitatea – medii diferite de rulare (hardware, SO, SGBD etc)
- ⋮

Constrângerile reprezintă acele cerințe non-funcționale care nu contribuie la îmbunătățirea arhitecturii și implicit a aplicației, dar care se aplică din diverse cauze și care pot influența decisiv arhitectura aplicației.

Ceea ce ne interesează este să găsim o soluție cât mai bună, atât din punct vedere funcțional, cât și calitativ, în condițiile impuse de constrângeri.

Tipuri de constrângeri

A. Fezabilitatea soluției → resurse ← timp ← termene → relevant și pentru proiectele din facultate (deadlines din proiect); ferestre de oportunitate
   ↳ oameni ← volum, calitate → relevante și pentru proiectele din facultate ← disponibilitate limitată; experiență limitată
B. Reglementări la nivel de domeniu (medical, bancar etc)
C. Competențe ← experiența de utilizare, tehnologie, cultură organizațională

buget (acoperă timp + oameni + altele ...)

Analiza arhitecturală - proiect

Tema: Realizați analiza arhitecturală pe modelul de mai jos pentru o aplicație software la care ați lucrat semestrul trecut sau lucrați semestrul acesta. Proiectul se poate realiza individual sau de către mai mulți studenți și studenți care au lucrat sau lucrează (tfi!) la aplicația respectivă.

Proces: Pasul 1: Înscrierea individuală într-un formular în echipa de curs din MS Teams, cu indicarea temei pe care vreți să o documentați și a colegilor, dacă veți face proiectul în echipă.

Pasul 2: Încărcarea proiectului de către unul dintre membrii echipei într-un Assignment în echipa de curs din MS Teams

Termene: Pentru Pasul 1: 3 zile lucrătoare de la data publicării formularului
Pentru Pasul 2: 10 zile lucrătoare de la data publicării Assignmentului.

Alte observații: a) Tema pe care o alegeți trebuie să fie suficient de complexă din punct de vedere arhitectural
b) În cazul unei echipe, realizarea proiectului va implica doar unele membrii echipei

Structura documentului și a proiectului în sine

1. Descrierea sumară a temei, care să conțină:
   a) Tipul de aplicație și categoria în care se încadrează
   b) Misiunea aplicației
   c) Modalitățile principale prin care vă propuneți să realizați misiunea aplicației. Ce este și ce nu este aplicația pe care urmează să o dezvoltați (delimitarea scopului).
2. Colectarea cerințelor – Etape
   Această secțiune are ca scop identificarea celor mai relevante cerințe. Ele se pot colecta fie de la stakeholderi fie din surse de documentare la care aveți acces

Foarte important: cerințele vor fi enumerate împreună cu sursa (sursele) din care provin

3. Ierarhizarea și prioritizarea cerințelor
   - eliminarea unor cerințe irelevante, reformularea și alinierea cerințelor și eliminarea redundanțelor
   - identificarea celor mai relevante cerințe din fiecare categorie (funcționale, de calitate, constrângeri)
     i) minim 3 cazuri de utilizare (cele mai reprezentative)
     ii) minim X criterii de calitate (cele mai importante), X = max 3, în documentație
     iii) un număr rezonabil de constrângeri relevante
4. Specificarea cerințelor selectate la Pasul 3, astfel:
   - Se pleacă de la misiunea aplicației, se scriu user story-urile principale care derivă direct din misiunea aplicației pe fiecare tip de cerință, și se descompun aceste user story-uri până la

Nivelul la care pot fi asociate cazurile de utilizare și scenariile de calitate corespunzătoare cerințelor selectate la pasul 3.

Analiza arhitecturală - proiect

Studiu de caz

1. Descrierea aplicației
   - tipul aplicației
   - misiunea
   - aplicația se rezumă doar la donații în bani
2. Colectare cerințelor
   - probleme cu cultura donatului în România
   - riscuri
   - oportunități

pentru realizarea misiunii

Pentru promotor – instrumente ajutătoare + automatizare

donator – campaniile de promovare
– redirecționarea unor taxe
– eveniment social la care se fac donații

admin – automatizare monitorizarea utilizatorilor și cazelor