Laborator 2 - Rolurile în echipă. Redactarea specificațiilor de proiect.

Rolurile în cadrul echipei de proiect

Cu cât proiectul este mai mare, cu atât rolurile din echipă sunt mai diversificate (mai specializate pe o anume arie de activități).

Într-un ***proiect mic***, principalele roluri din echipă sunt:

1. **Project Manager** / **Manager-ul Proiectului**
   - responsabil de succesul/eșecul proiectului
   - planifică & monitorizează evoluția proiectului
   - ia decizii
   - adesea, are background tehnic (pentru a înțelege ciclul de viață al proiectului)
   - întotdeauna, are abilități soft (de conducere, de motivare și stimulare, de conștientizare a cauzelor, de decizie, de negociere, de previziune, de inovație)
   - certificări PMI (Project Management Institute): PMP (Project Management Professional)
2. **Team Leader** / **Liderul echipei** (Dezvoltator Lider, QA Lider)
   - responsabil de execuția conform standardelor a activităților planificate
   - monitorizează activitatea echipei sale
   - intervine pentru a corecta, pentru a îndruma (coaching/support)
   - raportează managerului statusul real al proiectului
3. **Software Developer** / **Dezvoltator software**
   - responsabil de implementarea software a cerințelor
   - în metode de dezvolare agile contribuie și la design, arhitectură, specificații
4. **Tester** (Inginerul Calității)
   - responsabil de verificarea funcționalităților și a performanțelor produsului
   - scrie scenarii de test, le execută, analizează rezultatele ⇒ rapoarte de testare

Într-un ***proiect mare***, pe lângă rolurile de mai sus, apar și altele, precum:

1. **System Architect** / **Arhitectul Sistemului**
   - responsabil de arhitectura produsului software
   - captează interesele partenerilor de business și le transpune alegând arhitectura potrivită

2. **Technical Writer** / **Scriitorul tehnic**
   - responsabil de documentația tehnică a proiectului
   - primește documente tehnice de la ingineri, pe care le editează spre a fi corecte, clare și conforme cu standardele în vigoare
3. **Analyst** / **Analist** (Analist de Business, Analist de Sisteme Business, Analist de Sistem, Analist de Cerințe)
   - responsabil de înțelegerea corectă a cerințelor de business
   - studiază specificațiile clienților, clarifică toate detaliile proiectului cu clientul și cu partenerii (stakeholders)
   - redactează specificațiile aplicației software
4. **Consultant**
   - analizează fezabilitatea ofertelor de proiecte
   - propune soluții pentru diverse probleme legate de ciclul de viață al proiectelor
   - evaluează gradul de atingere a obiectivelor proiectelor
5. **Experți business**: SME (Subject Matters Expert), Manager-ul Produsului, Manager-ul de program

Gestiunea timpului

Orice membru al echipei și project manager-ul în special trebuie să aibă skill-uri de organizare a timpului pentru a obține maximum de rezultate. Aspecte precum planificarea task-urilor, stabilirea obiectivelor, evitarea întreruperilor, prioritizarea activităților trebuie avute constant în vedere pentru eficientizarea muncii.

Redactarea specificațiilor de proiect

După primirea specificațiilor clientului și după analiza de profunzime a acestora, contractorul redactează *documentele de specificație* ale soluției oferite, de obicei sub formă de:

- **Software Requirements Specification** / **Specificarea Cerințelor Software** (SRS/SCS)
- **Software Design Description** / **Descrierea Design-ului Software** (SDD/DDS)

Este **esențial** pentru reușita proiectului ca aceste documente de specificații să fie complete și absolut clare deoarece:

- acestea reprezintă parte a contractului dintre client și contractor;
- pe baza acestora se pot determina clar cerințele proiectului, rezolvându-se astfel eventualele conflicte dintre parteneri
- pe baza acestora se vor realiza estimările de timp și buget și planificările de proiect;

- pe baza lor se va realiza dezvoltarea soluției și testarea funcționalităților și performanțelor ei;
- pe baza lor se va realiza evaluarea finală a proiectului (a gradului de atingere a obiectivelor).

Documentul SRS cuprinde descrierea completă a comportamentului sistemului software, adică:

- a interacțiunilor dintre utilizator și sistem (**cerințe funcționale**). Acestea se descriu cu ajutorul *use-cases* (cazurilor de utilizare).
- a constrângerilor de proiectare și dezvoltare (**cerințe non-funcționale**). Acestea se referă la restricții de performanță, de securitate, de fiabilitate etc.

Conținutul unui SRS/SCS

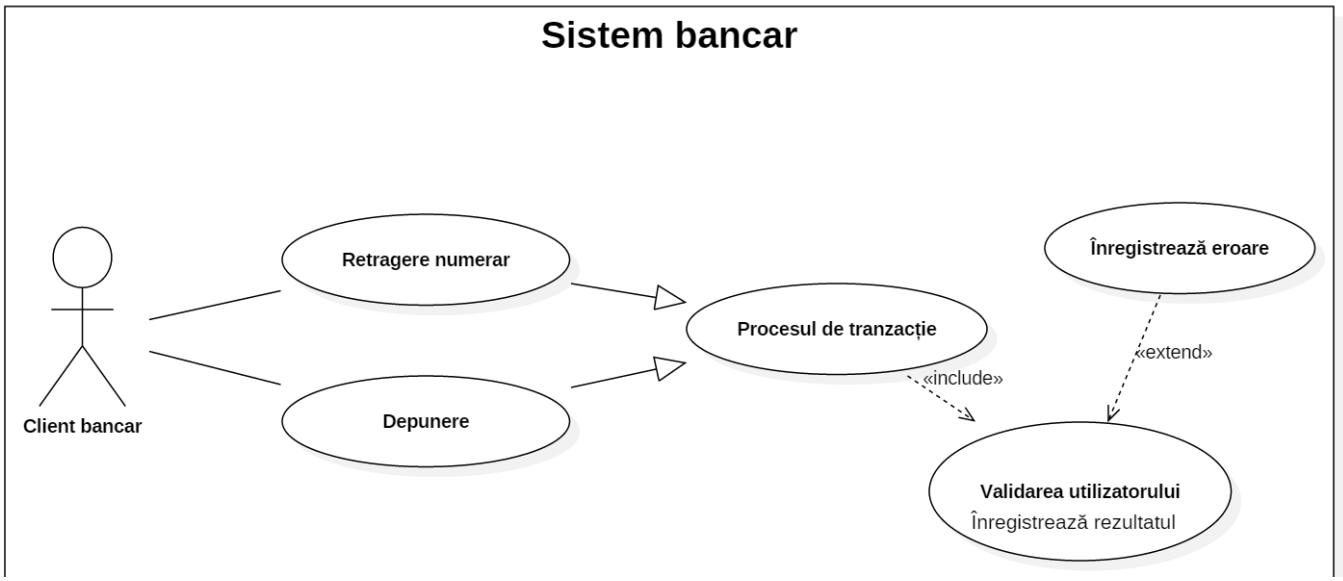
- Scopul documentului (*Document purpose*)
- Conținutul documentului (*Document overview*)
- Descrierea generală a produsului (***General description of the product***)
- Situația curentă (*The current situation*)
- Misiunea proiectului (*Purpose of the product*)
- Contextul proiectului (*Product context*)
- Beneficii (*Benefit*)
- Cerințe funcționale (***Functional requirements***)
- Actori (***Actors***)
- Diagrama de sistem (***System boundary***)
- Descrierea cazurilor de utilizare (***Use cases description***)
- Cerințe nefuncționale (***Non-functional requirements***)
- Cerințe de interfață (*User Interface Requirements*)
- Cerințe de performanță (*Performance Requirements*)
- Cerințe de fiabilitate (*Availability & Reliability*)
- Cerințe de securitate (*Security Requirements*)

System Boundary

Se mai numește și *Use-case diagram / Diagrama de cazuri de utilizare*.
Este:

- o diagramă UML obținută în urma analizei cerințelor utilizatorului
- schema funcționalităților oferite de sistemul software, în termeni de actori, cazuri de utilizare și relații între acestea.

Mai multe detalii, poți afla aici și aici.



Astfel de diagrame se pot realiza în Visio, Visual Paradigm sau Dia.

Exemplu de document SRS

Exemplu de Document SRS - Alumni database
Exemplu de Document SRS - Simulator cursa formula 1

Arhitectura proiectului

Stabilirea arhitecturii unui sistem software implică:

1. *alegerea șabloanelor arhitecturale* potrivite (se pot combina mai multe, se pot personaliza)
2. în funcție de șabloanele arhitecturale alese, *proiectarea structurii* sistemului software la nivel de:
   - **componente** specializate pe o gamă de servicii (module și interfețe),
   - **conexiuni** dintre componente.

Există mai multe tipuri șabloane arhitecturale (**architectural patterns**), precum:

- Tehnică de calcul distributivă
  - sistemul software e alcătuit dintr-o serie de componente software ce rulează pe mașini de calcul diferite, ce comunică prin rețea
  - mașinile de calcul interacționează pentru a oferi cât mai rapid un răspuns corect la cererea utilizatorului
- Client-server
  - arhitectură distribuită

    - conține componente furnizoare de servicii = componente *server* și componente ce solicită servicii = componente *client*
- Arhitectură bazată pe evenimente
    - se bazează pe producerea, detectarea, consumarea și reacția la evenimente
- Front-end și back-end
    - front-end = interfață de colectare a datelor de la utilizator și de procesare a acestora pentru a fi aduse la formatul prevăzut de componenta back-end
    - back-end = componenta software de procesare a datelor furnizate de front-end
- Modelul pe trei nivele
    - arhitectură de tip client-server
    - conține: nivel de prezentare (browser web), nivel de logică business (middleware - server de application: "motor" de rulare a paginilor web dinamice scrise în ASP.NET, JSP/Java, PHP, Perl, Python, Ruby), nivel de bază de date (server de bază de date)
    - browserul trimite cererile către serverul de aplicație, care le servește interogând și modificând conținutul bazei de date și, de asemenea, generează pagina de răspuns care va fi trimisă înapoi browserului.

Exemplu de descriere a arhitecturii unei aplicații: Platformă de gestiune a datelor medicale electronice

Software Design Document (SDD)

- document de **specificație** a soluției pentru sistemul software descris în **SRS**
- răspunde la întrebarea: *cum va fi construit sistemul software pentru a avea comportamentul descris în SRS?*
- prezinta metodologii, tehnologii, participanți și resurse implicate în proiect
- se poate redacta numai după finalizarea SRS-ului fiind un răspuns la cerințele prezentate în SRS
- este redactat de o echipă de software designers (proiectanți de sistem) și analiști business, pe baza SRS-ului și a experienței acestora
- reprezintă ghidul de construire a soluției folosit de echipa de dezvoltare a proiectului
- reprezintă un tool de analiză a întregului proiect în faza de început cât și de monitorizare pe parcurs

Secțiuni SDD

Un SDD are următoarele secțiuni:

1. **Scopul documentului (Document purpose)**
2. **Obiective (Objectives)**
3. **Conținutul documentului (Document overview)**
4. **Modelul datelor** (*Data Design*)
   - prezintă *structurile de date* importante, *formatele fișierelor* folosite în cadrul soluției și *schema bazei de date*.
   - Structurile de date pot fi:
     - **globale** (structuri de date disponibile tuturor componentelor arhitecturii)
     - **de legătură** (structuri de date trimise ca argumente între componente pentru a asigura fluxul informației la nivel de aplicație)
     - **temporare** (structuri de date folosite temporar).
   - Schema bazei de date este descrisă prin:
     - **diagrama schemei** bazei de date
     - **descrierea semnificației tabelelor** și, pentru fiecare tabelă, descrierea semnificației coloanelor și indicarea cheilor primare și referențiale.
5. **Modelul arhitectural** (*Architectural Design*)
   - este o structură ierarhică alcătuită din componente interconectate
   - prezintă arhitectura sistemului – atât descriptiv, cât și sub forma unei diagrame de arhitectură
   - descrie:
     - fiecare componentă a arhitecturii,
     - restricțiile de implementare ale componentelor,
     - interacțiunea dintre componentele sistemului.
   - poate fi reprezentat prin :
     - **diagrame de componente** (pentru proiecte mari) - le-am numit "generic" în laboratorul 2: diagrame de arhitectură
     - **diagrame de clase**, în care relațiile ierarhice se bazează pe generalizare și specializare (pentru proiecte mici).
6. **Modelul interfeței cu utilizatorul** (*User Interface Design*)
   - prezintă *ferestrele* aplicației și *succesiunea* lor în cadrul sistemului.
7. **Elementele de testare** (*Testing Issues*)
   - identifică *componentele critice* ale aplicației (componente a căror performanță influențează decisiv performanța globală a aplicației)

- propune *alternative* de proiectare a componentelor critice (pentru a fi folosite în cazul insuccesului soluției propuse inițial).

Exemplu de Document SDD

Exemplu de document SDD mai vechi: Exemplu de Document SDD
Exemplele de mai jos sunt mai noi. Unul se remarcă prin organizarea clară a informaţiilor şi structură/organizare, dar şi prin concizie, însă Scopul şi Obiectivele nu sunt în totalitate corecte. Cel de-al doilea se remarcă prin forma simplă şi schematică, dar care include toate secţiunile importante (şi chiar extra) într-un mod concis.

Exemple mai noi