



PROIECT FINAL

TIMOFTE ALEXANDRU BOGDAN

DATA EXAMEN 10.04.2025

INTRODUCERE

Există două discipline diferite implicate în testarea software: **testarea manuală** și **testarea automată**. În ciuda faptului că ambele au efectiv aceeași funcție, ele sunt discipline distincte pe care companiile le folosesc pentru a-și examina pachetele software.

- Procesul de testare software este structurat pe **patru niveluri de testare**, fiecare având un scop distinct. Acestea sunt:

- 1. Testarea Unităților** – verifică funcții/module individuale.
 - 2. Testarea de Integrare** – verifică interacțiunea dintre module.
 - 3. Testarea Sistemului** – testează produsul ca întreg.
 - 4. Testarea de Acceptanță** – validează că produsul este gata pentru utilizatori.
- Fiecare nivel contribuie la identificarea și prevenirea defectelor, asigurând livrarea unui produs de calitate.

- Evaluarea cerințelor de business are loc în **etapa de revizuire a cerințelor** sau **testarea statică**, care este o activitate inițială în cadrul procesului de testare software. Aceasta face parte din **Testarea de Validare** și se realizează înainte de implementarea efectivă a codului.

- **Etapele principale unde sunt evaluate cerințele de business:**

1. Analiza cerințelor – Se verifică dacă cerințele sunt complete, corecte, fezabile și testabile.

2. Revizuirea documentației – Se analizează specificațiile funcționale și non-funcționale pentru a identifica eventuale ambiguități sau contradicții.

3. Testarea statică – Se utilizează tehnici precum **revizuirea documentelor, walkthrough, inspecții** pentru a identifica probleme înainte de implementare.

- Această evaluare timpurie ajută la prevenirea defectelor și reduce costurile corectării erorilor în etapele ulterioare de dezvoltare și testare.

- **Criteriile de intrare și criteriile de ieșire** sunt condiții utilizate în procesul de testare software pentru a determina când poate începe sau când se poate considera finalizată o activitate de testare.

- **Criteriile de intrare** definesc **ce trebuie să fie pregătit** pentru a începe testarea.
- **Criteriile de ieșire** definesc **ce trebuie să fie îndeplinit** pentru a încheia testarea.

- **Precondition (Precondiție)**

O precondiție este o condiție care trebuie să fie îndeplinită înainte de executarea unui test. Practic, reprezintă starea inițială necesară pentru ca testul să fie valid.

- **Exemple de precondiții:**

- Utilizatorul trebuie să fie autentificat înainte de a testa funcționalitatea de schimbare a parolei.
- Baza de date trebuie să conțină un produs înainte de a testa procesul de adăugare în coș.
- Serverul de test trebuie să fie activ înainte de a începe testele de performanță.

- **Test Condition (Condiție de testare)**

O condiție de testare este un aspect specific al sistemului care trebuie verificat în cadrul testării. Acesta este un element derivat din cerințele software și definește **ce trebuie testat**.

- **Exemple de condiții de testare:**

- Verificarea dacă un utilizator poate reseta parola.
- Asigurarea că utilizatorul primește un mesaj de eroare dacă introduce date incorecte.
- Testarea funcționalității de adăugare a unui produs în coșul de cumpărături.
- **Precondiția este o stare necesară înainte de a rula testul.**
- **Condiția de testare este ceea ce urmează să fie verificat în cadrul testului.**

- In procesul de testare se vor folosi diferite tool-uri pentru a ne face munca mai usoara si de a simplifica trecerea de informatii de la un departament la altul.

- Un Instrument Esențial pentru Managementul Proiectelor este **JIRA.**

- **Jira** este o platformă de management al proiectelor dezvoltată de Atlassian, utilizată în principal pentru urmărirea erorilor și gestionarea activităților în cadrul echipelor de dezvoltare software. Datorită flexibilității și adaptabilității sale, Jira este folosită și în alte domenii, precum IT, marketing și gestionarea proceselor de afaceri.

- **Istoricul și Evoluția** Jira Lansată în 2002 de compania Atlassian, Jira a evoluat de-a lungul anilor pentru a răspunde nevoilor complexe ale echipelor de dezvoltare. Inițial concepută ca un instrument de urmărire a erorilor, Jira a devenit o soluție completă pentru gestionarea proiectelor agile, fiind integrată cu diverse metodologii precum Scrum și Kanban.

- **Principalele Caracteristici ale Jira:**

1.Urmărirea Task-urilor și Bug-urilor – Jira permite echipelor să creeze, să asigneze și să monitorizeze task-uri și erori pe parcursul ciclului de viață al proiectului.

2.Metodologii Agile – Suportă framework-uri precum Scrum și Kanban, facilitând planificarea sprinturilor, gestionarea backlog-ului și analiza progresului.

3.Automatizare și Workflows Personalizabile – Jira oferă posibilitatea de a defini fluxuri de lucru personalizate, optimizând astfel gestionarea proceselor și reducerea timpului de execuție.

4.Raportare și Analiză – Include instrumente avansate de raportare, cum ar fi burndown charts, velocity charts și rapoarte personalizabile pentru a evalua performanța echipei.

5.Integrare cu Alte Instrumente – Jira se poate conecta cu aplicații precum Confluence, Bitbucket, Trello și Slack pentru o colaborare mai eficientă.

6.Managementul Drepturilor de Acces – Oferă posibilitatea de a defini roluri și permisiuni pentru utilizatori, asigurând controlul accesului la diferite funcționalități.

7.Personalizare și Extensibilitate – Jira permite adăugarea de plugin-uri și configurarea tablourilor de bord pentru a se adapta nevoilor specifice ale fiecărei echipe.

8.Suport pentru DevOps – Se integrează cu instrumente CI/CD, facilitând livrarea continuă a software-ului și monitorizarea performanței.

- Un **release** reprezintă versiunea finală a unui produs software sau a unei actualizări, care este livrată utilizatorilor finali. Acesta conține noi funcționalități, îmbunătățiri sau remedieri de bug-uri și poate fi distribuit intern (echipe de testare, clienți beta) sau public.
- **Cycle Summary** este un raport care detaliază rezultatele unui ciclu de testare și ajută la luarea deciziei de release.
- Un *release* este despre **distribuirea produsului**, iar un *cycle summary* este despre **evaluarea testării** înainte de release.

- Cei doi parametri principali după care se evaluează importanța unui **bug** sunt:

1. Severitatea reflectă **gravitatea defectului** din punct de vedere tehnic.

2. Prioritatea reflectă **urgenta și importanța** rezolvării defectului din perspectiva utilizatorului și business-ului.

- ***Test Status Report* și *Test Summary Report*** sunt folosite pentru a urmări progresul testării, dar ele servesc scopuri diferite și conțin informații diferite.

• **Test Status Report** este un raport **intermediar** care reflectă starea curentă a testării și progresele realizate până în acel moment.

• **Test Summary Report** este un raport **final** care oferă un rezumat al întregului proces de testare, inclusiv concluzii, defecte descoperite și recomandări pentru următoarele etape.

- **Matricea de trasabilitate** este un document sau un instrument utilizat în managementul testării software pentru a urmări și a demonstra legătura dintre cerințele de business și testele efectuate. Ea asigură că toate cerințele au fost testate corespunzător și ajută echipele de testare să verifice că fiecare cerință este acoperită printr-un caz de testare adecvat.

- **Utilitatea Matricei de Trasabilitate:**

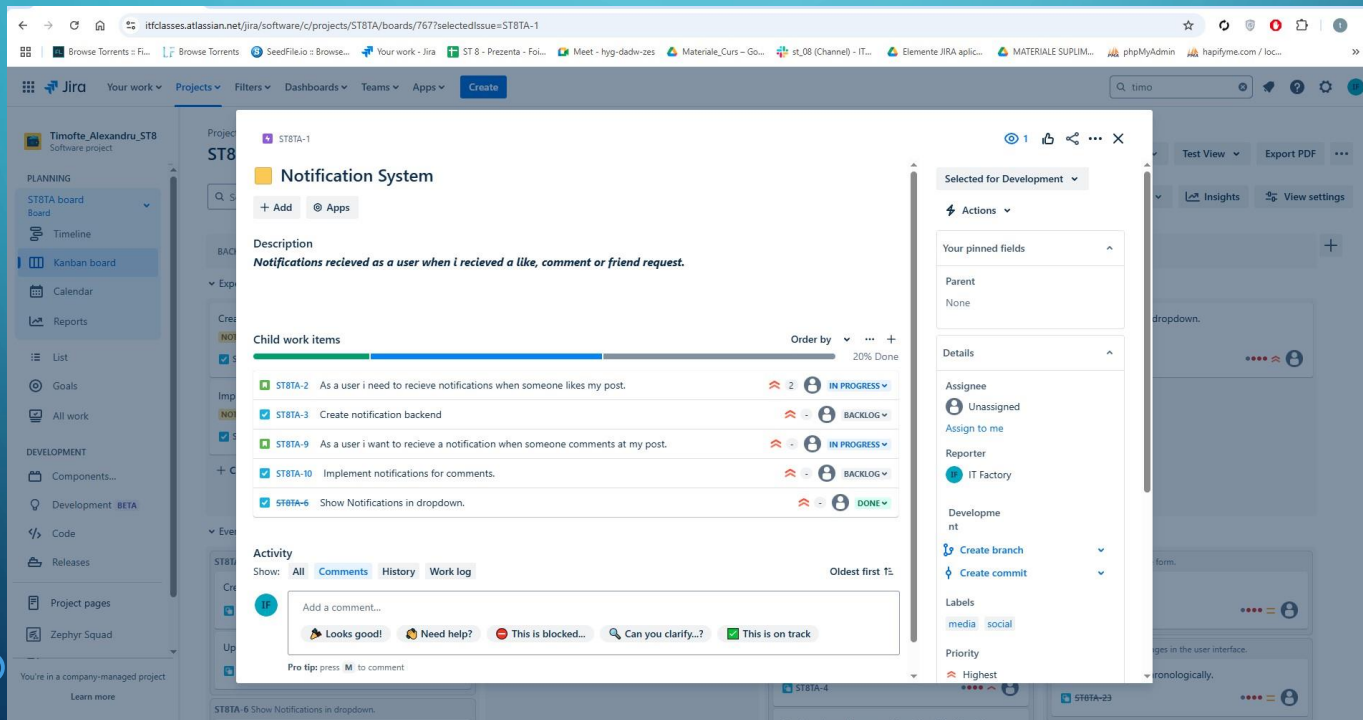
1. **Asigură acoperirea completă a cerințelor**
2. **Ușurează gestionarea schimbărilor**
3. **Transparență pentru părțile interesate**
4. **Verificarea conformității cu cerințele de business**
5. **Reducerea riscurilor**

Proiect Jira

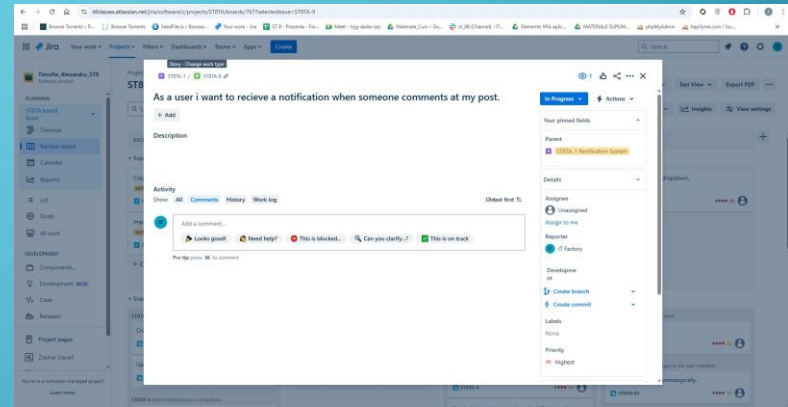
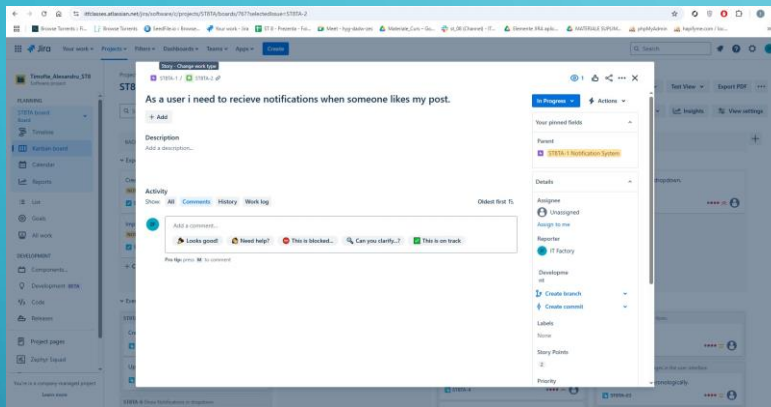
În Jira, structura ierarhică a elementelor de lucru (issues) este organizată astfel:

1.Epic – O unitate mare de lucru care conține mai multe Stories, Tasks sau Bugs. Reprezintă o funcționalitate majoră sau un obiectiv mare.

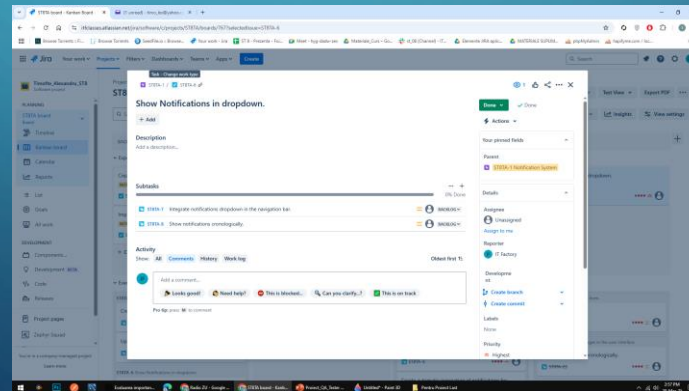
De exemplu, în această situație este creat pentru a rezolva o problemă majoră și bineînțeles urgentă. Utilizatorul nu primește notificările atunci când primește un like, coment sau cerere de prietenie.



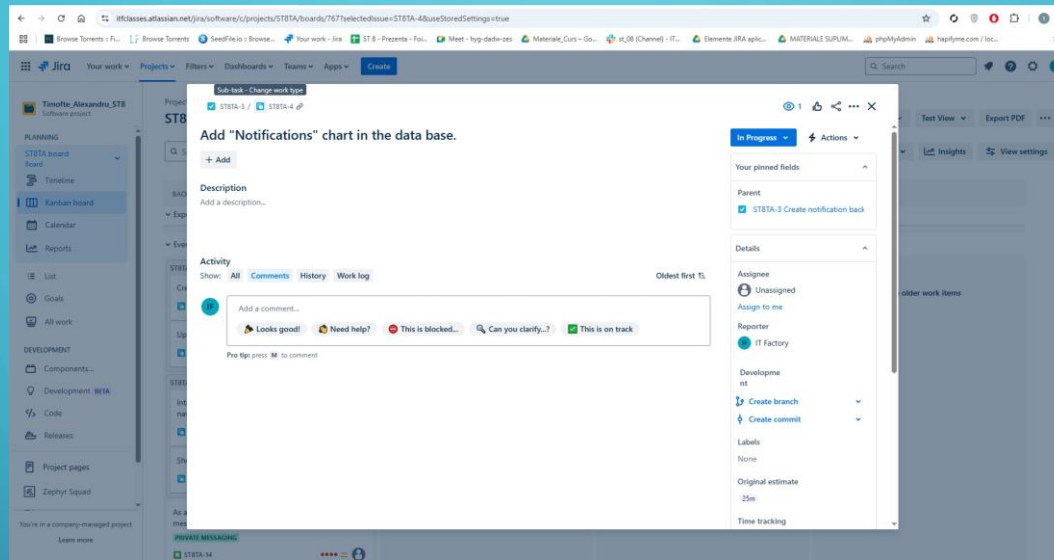
2.Story – O cerință funcțională sau o caracteristică care trebuie implementată. Este o unitate de lucru mai mică dintr-un Epic. In aceasta situatie , din Epic-ul “Notification System” se creeaza doua subnivele pentru a rezolva problema mult mai usor.



3.Task – O sarcină independentă care trebuie realizată. Poate face parte dintr-un Epic sau poate fi un element de sine stătător.



4.Sub-task – O subdiviziune a unui Task sau Story, utilizată pentru a împărți lucrul în părți mai mici.



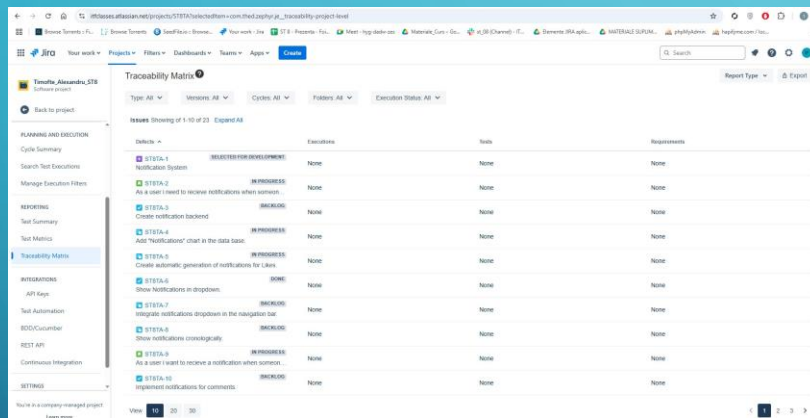
5.Bug – O eroare sau problemă care trebuie remediată. Poate fi asociat cu un Story, Task sau Epic.

6.Spike – Un task special utilizat pentru cercetare sau investigație, folosit adesea pentru a clarifica cerințele înainte de dezvoltare.

Această ierarhie ajută echipele să organizeze eficient proiectele și să urmărească progresul fiecărei componente.

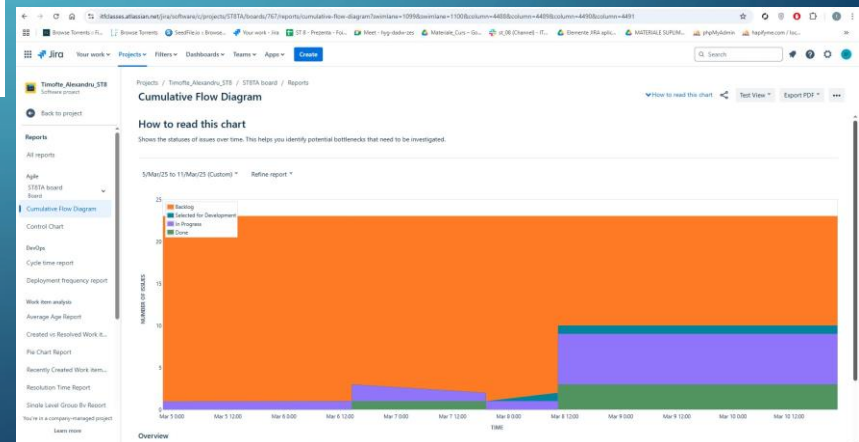
Tipuri de Traceability Matrix:

- 1. Forward Traceability** – Urmărește cerințele și asigură că fiecare cerință are un caz de testare asociat.
- 2. Backward Traceability** – Verifică dacă fiecare caz de testare este legat de o cerință, evitând testarea inutilă.
- 3. Bidirectional Traceability** – Oferă trasabilitate în ambele direcții, asigurându-se că toate cerințele sunt testate și că fiecare test corespunde unei cerințe.



The screenshot shows the JIRA Traceability Matrix for a project named 'Timothy_Alexander_STB'. The table displays requirements and their associated test cases, categorized by status (To Do, In Progress, Done).

Requirements	Test Cases	Test Case Status
STB1.1: Notification System	STB1.1.1: As a user I want to receive notifications when someone...	To Do
STB1.2: Add 'notifications' chat in the data base	STB1.2.1: Create automatic generation of notifications for Users	In Progress
STB1.3: Show notifications in dropdown	STB1.3.1: Integrate notifications dropdown in the navigator bar	In Progress
STB1.4: Show notification consistency	STB1.4.1: As a user I want to receive a notification when someone...	In Progress
STB1.5: Integrate notifications for comments	STB1.5.1: Integrate notifications for comments	In Progress



The background is a blue gradient with faint concentric circles. White circuit-like lines with circular nodes are positioned in the corners: top-left, top-right, bottom-left, and bottom-right.

Mulțumesc pentru atenție!