**Cerințe Managementul si organizarea proiectelor software**

Notarea se va face pe baza proiectului de laborator (50%) si proiectului de curs (50%). **Este important ca proiectele să fie originale.** (v. [Regulamentul de etică și profesionalism al FMI](https://drive.google.com/file/d/1gw2Fy44KnqaBKqLGh70vZPo3T9-ZdrDi/view) - pct. 11 de la pagina 3 despre incidente minore).

1. **Proiect de laborator**

**Dimensiunea recomandată a echipei** pentru acest proiect **este de 4-5 studenți**. În acest proiect echipa va dezvolta **parțial** o aplicație (cerințele sunt la alegerea echipei) folosind **metodologia SCRUM** [1,21]. Dezvoltarea va consta dintr-un singur Sprint, având 5 zile lucrătoare (40 de ore). Se recomandă ca aceste 5 zile sa fie distribuite de-a lungul a 5 săptămâni (o zi / săptămână).

Proiectul trebuie să fie complex și inovativ. Cadrul didactic de la laborator vă va ghida în alegerea proiectului.

**Roluri**: Unul dintre membrii echipei își va asuma rolul de Product Owner, ceilalți fiind Dezvoltatori. Dintre Dezvoltatori, unul iși va asuma simultan și rolul de Scrum Master. În echipă nu vor exista Testeri dedicați, testarea unitară fiind inclusă în procesul de implementare, iar testarea de acceptanță va fi făcută încrucișat (de către un alt Dezvoltator decât cel care a realizat implementarea).

**Cerințe:**

1. Se pornește de la un scenariu propriu care descrie cerințele sistemului ce urmează a fi dezvoltat (Requirements Document). Acesta va fi însoțit de o Diagramă de Cazuri de Utilizare (Use Case Diagram) [1,25,26]. (Responsabilitate: Product Owner)
2. Din Requirements Document vor fi derivate User Stories inițiale [1-5]. (Notă: cerințele sistemului vor fi alese astfel încât să rezulte cel puţin 15 User Stories.) (Responsabilitate: Product Owner)
3. User Stories de la punctul anterior vor fi rafinate astfel încât să satisfacă cele 6 criterii: Independent, Negotiable, Valuable to users or customers, Estimable, Small, Testable (INVEST) [1-3]. Să se justifice modificările făcute. Fiecărei User Story i se va atribui o prioritate, care va fi justificată. User Stories rezultate vor alcătui Project Backlog. (Responsabilitate: Product Owner și Dezvoltatorii)
4. Să se estimeze efortul (în story points) pentru fiecare User Story folosind Planning Poker [7,8] și să se justifice estimarea. (Responsabilitate: Dezvoltatori)
5. Se completeaza Sprint Backlog in Planning Meeting (care are loc in prima zi a Spint-ului). Observatie: În mod normal, **Sprint Backlog nu va cuprinde toate User Stories din Project Backlog;** din contră, având în vedere durata redusă a sprint-ului și dimensiunea echipei, este de așteptat ca doar o parte din ele să poată fi incluse în Sprint Backlog, selecția fiind făcută în funcție de estimarea efortului și priorități. (Responsabilitate: Dezvoltatori, Scrum Master, Product Owner)
6. Se va folosi un tool (la alegerea echipei) pentru project management și issue tracking [5,12]. (Responsabilitate: Dezvoltatori, Scrum Master, Product Owner)
7. Scrum Master-ul, cu ajutorul dezvoltatorilor, va realiza urmatoarele diagrame (aceste diagrame vor fi incluse în Project Backlog):
   1. Diagrama de workflow a proiectului.
   2. Diagrama bazei de date (dacă este cazul).
8. Product Owner-ul, cu ajutorul întregii echipe va dezvolta urmatoarele diagrame (aceste diagrame vor fi incluse în Project Backlog):
   1. Diagrama UML de cazuri de utilizare.
   2. Diagrama Gantt
9. Să se implementeze User Stories:
   1. aproximativ jumătate dintre acestea folosind metoda **Test-Driven Developmen**t [1,19].
   2. iar restul in mod traditional, însă folosind tehnica **Pair Programming** [22,23].
   3. Să se descrie procesul şi evoluţia testelor.
   4. Să se scrie cazurile de testare rezultate folosind un utilitar adecvat (e.g. JUnit pentru limbajul Java [17]).
   5. Să se stabileasca nivelul de acoperire realizat de fiecare dintre seturile de teste folosind un utilitar de code coverage precum cele prezentate in [20].
   6. Să se comenteze avantajele si dezavantajele Test-Driven Development față de dezvoltarea tradițională.
   7. Să se comenteze avantajele si dezavantajele Pair Programmingfață de programarea individuală.

(Notă: limbajul de programare este ales de echipa astfel încât să fie adecvat sistemului dezvoltat şi cerinţelor proiectului. Responsabilitate: Dezvoltatori).

1. Să se utilizeze un sistem de control al versiunilor, de exemplu SVN (Subversion), CVS (Concurrent Version System), GIT etc [13,14]. Să se folosească o platformă (cu găzduire free) pentru repository-ul proiectului dezvoltat, să se instaleze şi utilizeze diferiți clienți pentru sistemul de control al versiunilor ales (de exemplu SmartSVN, TurtoiseSVN, Subclipse pentru Subversion). (Responsabilitate: Dezvoltatori, Scrum Master)
2. Sprint Daily Standup Meetings vor avea loc în fiecare zi din Sprint, iar raportul proiectului va consemna principalele impedimente apărute. (Responsabilitate: întreaga echipa)
3. Implementarea (fiecărei User Story în parte) va fi prezentată (și validată/ invalidată de către Product Owner) în Review Meeting, care va avea loc in ultima zi a Sprint-ului. Tot în ultima zi va avea loc și o foarte scurtă (maxim 20 minute) Retrospective Meeting. (Responsabilitate: întreaga echipa, facilitată de Scrum Master)
4. Pentru comunicare intre membrii echipei se va folosi Slack [24].
5. Să se precizeze (ca procentaj, la sfârșitul raportului) contribuția fiecărui membru al echipei la realizarea proiectului.

**Evaluare**

Evaluarea acestui proiect se va face la laborator: cadrul didactic de laborator va stabili obiectivele de etapă (**project milestones)**, evaluarea fiind făcută la sfârșitul fiecărei etape.

**Termene:**

1. până pe 1 noiembrie: după o lună de la inceputul semestrului, ați fixat echipa și cerințele/backlogul proiectului (cadrul didactic de laborator își dă acordul)
2. până pe 1 decembrie: ați implementat cel putin 1/3 din backlog (cadrul didactic de laborator verifică și renegociați schimbări ale backlogului, dacă e cazul)
3. până pe 22 decemrbie: ați implementat cel puțin 2/3 din backlog (cadrul didactic de laborator verifică și renegociați schimbări ale backlogului, dacă e cazul)
4. până pe 15 februarie: prezentați întreg proiectul (cadrul didactic de laborator verifică dacă a fost implementat tot backlogul) și primiți nota pe proiect de la cadrul didactic de laborator.

Pentru fiecare checkpoint care nu este prezentat satisfăcător, pierdeți 0,75 pct din nota finală la laborator.

Prezența individuală la laborator: daca ați participat la mai puțin de 3 laboratoare din cele 7, pierdeți 2 pct din cele de mai sus. Dacă ați participat la toate cele 7 laboratoare, caștigați un punct în plus.

**Bibliografie**

1. Florentin Ipate: note de curs (Scrum, XP, TDD, Use Case Model).
2. Mike Cohn, User Stories Applied, Addison-Wesley (Chapter 2), <http://athena.ecs.csus.edu/~buckley/CSc191/User-Stories-Applied-Mike-Cohn.pdf>
3. <https://www.mountaingoatsoftware.com/uploads/presentations/User-Stories-Applied-Agile-Software-Development-XP-Atlanta-2004.pdf>
4. <https://www.romanpichler.com/blog/10-tips-writing-good-user-stories/>
5. <https://www.atlassian.com/agile/project-management/user-stories>
6. <https://www.adaptavist.com/blog/how-to-create-an-effective-jira-user-story>
7. <https://www.7pace.com/blog/user-story-estimation-techniques>
8. <https://www.visual-paradigm.com/scrum/what-is-agile-planning-poker/>
9. <https://www.sciencedirect.com/science/article/pii/S0164121212001021>
10. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0ae75cc0678f6b01f1066093b19cad9dd4379385>
11. <https://is.muni.cz/th/pbpa1/thesis_gyyik.pdf>
12. <https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems>
13. <https://www.sciencedirect.com/science/article/pii/S1877050918314819>
14. <https://www.atlassian.com/git/tutorials/what-is-version-control>
15. <https://arxiv.org/abs/2004.10777>
16. <https://refactoring.guru/refactoring/techniques>
17. <https://www.browserstack.com/guide/junit-and-automation-testing>
18. [https://www.jetbrains.com/help/idea/tutorial-introduction-to-refactoring.html#f6c540f0](https://www.jetbrains.com/help/idea/tutorial-introduction-to-refactoring.html%23f6c540f0)
19. [https://books.google.ro/books?hl=en&lr=&id=CUlsAQAAQBAJ&oi=fnd&pg=PR7&dq=test+driven+development&ots=QChW467PRX&sig=CD-L\_nqaKVpL8lvXtKCjt7z-D\_c&redir\_esc=y#v=onepage&q=test%20driven%20development&f=false](https://books.google.ro/books?hl=en&lr=&id=CUlsAQAAQBAJ&oi=fnd&pg=PR7&dq=test+driven+development&ots=QChW467PRX&sig=CD-L_nqaKVpL8lvXtKCjt7z-D_c&redir_esc=y%23v=onepage&q=test%252520driven%252520development&f=false)
20. <https://www.softwaretestinghelp.com/code-coverage-tools/>
21. <https://theagiledirector.com/images/IntroductiontoScrum-coursenotes.pdf>
22. <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/XPSardinia.pdf>
23. https://files.spiderboy.fr/madchat/coding/xp/xpexplored.pdf
24. https://slack.com/intl/en-gb/
25. Martin Fowler: UML Distilled: A Brief Guide to the Standard Object Modeling Language.
26. https://csis.pace.edu/~marchese/CS389/L9/Use%20Case%20Diagrams.pdf
27. **Proiect de curs**

**Dimensiunea recomandată a echipei** pentru acest proiect **este de 2 studenți** (și nu poate depăși 3 studenți, caz în care dimensiunea sporită a echipei va fi justificată de o complexitate mai mare a proiectului).

Aceste proiectele vor fi prezentate la curs in saptamanile 4-12 conform programarii stabilite, astfel incat numarul maxim de studenti care prezinta in cadrul unui curs sa fie 10.

Tema proiectului de curs va fi aleasa din lista de mai jos. Fiecare tema din lista poate fi aleasa de cel mult 3 echipe.

Prezentarea proiectului de curs va fi sub forma **de slide-uri, insotite de demo-uri**. La prezentare este necesara prezenta intregii echipe, fiecare student descriind principala sa contributie la proiect. Timpul alocat fiecarui student este de **10 minute**. Atunci cand echipa considera ca nu toti membrii echipei au avut o contributie egala la realizarea proiectului, se va indica, in procente, contributia estimata a fiecaruia.

**Teme proiecte curs**

1. **GitHub Copilot**

Realizați o prezentare despre utilizarea GitHub Copilot pentru îmbunătățirea procesului de dezvoltare software. Includeți o demonstrație despre cum Copilot asistă la completarea codului, generează fragmente de cod și automatizează sarcinile de codare de rutină. Discutați beneficiile și limitările utilizării Copilot, susținute de exemple din experiența voastră, și oferiți sugestii despre cum poate îmbunătăți productivitatea și practicile de codare.

2. **GitHub Merge Conflict Helper**

Realizați o prezentare despre utilizarea **GitHub Merge Conflict Helper** pentru rezolvarea conflictelor de "merge" în proiectele software colaborative. Demonstrați cum instrumentul identifică schimbările conflictuale, ghidează dezvoltatorii prin procesul de rezolvare a conflictelor și se integrează în fluxul de lucru tipic al GitHub. Evidențiați beneficiile și posibilele provocări ale utilizării Merge Conflict Helper, susținute de exemple și sfaturi practice din experiența voastră.

3. **Sourcery**

Realizați o prezentare despre utilizarea **Sourcery** pentru îmbunătățirea calității codului prin refactorizare automată. Demonstrați cum se integrează Sourcery cu mediile de dezvoltare, identifică oportunități de îmbunătățire a codului și oferă sugestii pentru un cod mai curat și mai eficient. Discutați beneficiile utilizării Sourcery, cum ar fi productivitatea crescută și lizibilitatea codului, precum și orice limitări sau provocări întâlnite.

4. **DeepCode**

Realizați o prezentare despre utilizarea **DeepCode** pentru îmbunătățirea securității și calității codului prin analiza codului bazată pe AI. Demonstrați cum se integrează DeepCode în fluxurile de lucru de dezvoltare, detectează vulnerabilitățile și oferă sugestii practice pentru îmbunătățirea codului. Discută avantajele utilizării DeepCode, cum ar fi securitatea îmbunătățită și reducerea ratei de erori, precum și orice limitări sau provocări întâmpinate în timpul utilizării acestuia.

5. **Jira** **and its Automation features**

Realizați o prezentare despre utilizarea **Jira** și a funcțiilor de **automatizare din Jira** pentru a optimiza gestionarea proiectelor și a îmbunătăți eficiența fluxurilor de lucru. Demonstrați cum poate fi folosit Jira pentru a gestiona sarcinile, a urmări progresul proiectelor și cum pot fi configurate regulile de automatizare pentru a automatiza sarcinile repetitive, cum ar fi atribuirea automată a problemelor, actualizarea stărilor sau trimiterea de notificări. Discutați beneficiile utilizării automatizării în Jira, cum ar fi economisirea timpului și reducerea erorilor manuale, precum și orice provocări întâmpinate la configurarea și utilizarea acestor automatizări.

6. **Ponicode**

Realizați o prezentare despre utilizarea **Ponicode** pentru automatizarea generării de teste unitare în dezvoltarea software. Demonstați cum se integrează Ponicode cu mediile de dezvoltare, cum generează cazuri de testare și cum ajută la îmbunătățirea acoperirii testelor și a calității codului. Discutați beneficiile utilizării Ponicode, cum ar fi eficiența crescută și fiabilitatea în testare, precum și orice limitări sau provocări întâmpinate în timpul utilizării acestuia.

7. **Trello** **with Butler Automation**

Realizați o prezentare despre utilizarea **Trello** cu **Butler Automation** pentru a îmbunătăți gestionarea sarcinilor și eficiența fluxurilor de lucru. Demonstrați cum Butler poate automatiza sarcini comune în Trello, cum ar fi mutarea cardurilor, setarea termenelor limită și trimiterea de reminder-e pe baza unor declanșatori și reguli specifice. Discutați beneficiile utilizării automatizării cu Butler pentru a economisi timp și a reduce efortul manual, precum și orice limitări sau provocări întâmpinate la configurarea acestor automatizări.

8. **Monday.com and its AI-powered automation features**

Realizați o prezentare despre utilizarea **Monday.com** și a funcțiilor de **automatizare bazate pe AI** pentru a îmbunătăți gestionarea proiectelor și a eficientiza fluxurile de lucru. Demonstrați cum se configurează automatizările în Monday.com pentru a atribui automat sarcini, a actualiza stările și a trimite notificări pe baza unor declanșatori și condiții specifice. Discutați avantajele utilizării automatizării bazate pe AI pentru a crește productivitatea și a reduce erorile manuale, precum și orice provocări sau limitări întâmpinate la implementarea acestor funcții.

9. **Diffblue Cover**

Realizați o prezentare despre utilizarea **Diffblue Cover** pentru automatizarea generării de teste unitare pentru aplicațiile Java. Demonstrați cum se integrează Diffblue Cover într-un mediu de dezvoltare, cum se generează teste unitare și evaluați acoperirea și eficacitatea acestor teste. Discutați avantajele utilizării Diffblue Cover, cum ar fi îmbunătățirea calității codului și reducerea efortului de testare manuală, precum și orice limitări sau provocări întâmpinate în timpul implementării acestuia.

10. **IntelliJ IDEA**’**s AI features**

Realizați o prezentare despre utilizarea **funcțiilor AI din IntelliJ IDEA** pentru a spori productivitatea în dezvoltarea software și calitatea codului. Demonstrați cum se utilizează funcții precum completarea inteligentă a codului, analiza codului și refactorizarea automată oferite de capabilitățile AI ale IntelliJ. Discutați beneficiile acestor instrumente bazate pe AI în ceea ce privește accelerarea dezvoltării, reducerea erorilor și îmbunătățirea lizibilității codului, precum și orice limitări sau provocări întâmpinate în utilizarea acestora.

11. **TestNG**

Realizați o prezentare despre utilizarea **TestNG** pentru implementarea testării automate în aplicațiile Java. Demonstrați cum se integrează TestNG într-un mediu de dezvoltare, cum se creează cazuri de testare folosind adnotări și cum se execută testele pentru a verifica funcționalitatea. Discutați beneficiile utilizării TestNG, cum ar fi suportul pentru execuția în paralel, gruparea testelor și testarea bazată pe date, precum și orice provocări sau limitări întâmpinate în utilizarea framework-ului.

12. **TabNine**

Realizați o prezentare despre utilizarea **TabNine** pentru a îmbunătăți eficiența codului și a reduce erorile în dezvoltarea software. Demonstrați cum se integrează TabNine cu IDE-urile populare, cum evidențiază funcțiile sale de completare a codului bazate pe AI și cum oferă sugestii dependente de context în timp real. Discutați beneficiile utilizării TabNine, cum ar fi creșterea productivității și consistenței în codare, precum și orice limitări sau provocări întâmpinate în utilizarea acestui instrument.