Universitatea Politehnica Timișoara

MANAGEMENTUL PROIECTELOR SOFTWARE

TEHNICI PRACTICE ÎN METODOLOGIA AGILE

PROIECT REALIZAT DE: Giurea Laura-Maria
PROFESOR COORDONATOR: Bocan Valer

CUPRINS

1. INTRODUCERE

Agile este o metodologie de dezvoltare software care se concentrează pe livrarea rapidă și incrementală de soluții, punând accentul pe adaptarea eficientă la schimbările cerințelor și feedback-ul clienților pe parcursul întregului proces de dezvoltare. Metodologiile Agile promovează colaborarea strânsă între dezvoltatori și clienți, încurajând interacțiunea directă și comunicarea continuă pentru a asigura că produsul final reflectă cu precizie cerințele și așteptările utilizatorilor. Metodologia Agile are la bază "Manifestul pentru dezvoltarea agilă de software", un set de principii și valori fundamentale cu privire la dezvoltarea și gestionarea proiectelor software.

Unul dintre aspectele cheie care contribuie la succesul proiectelor ce respectă principiile Agile îl reprezintă practicile tehnice încorporate în procesul de dezvoltare. În acest context, proiectul meu se concentrează pe explorarea a patru practici tehnice fundamentale în metodologia Agile: test-driven development, refactorizarea, simple design și pair programming. Aceste practici nu numai că îmbunătățesc calitatea codului și productivitatea echipei, dar contribuie și la flexibilitatea și adaptabilitatea proiectului în fața schimbărilor și cerințelor în continuă evoluție.

2. STADIUL ACTUAL ÎN DOMENIU

Înainte de introducerea metodologiei Agile, majoritatea proiectelor software erau dezvoltate pe baza metodologiei "Waterfall" (cascadă), care necesita respectarea cerințelor inițiale ale proiectului, astfel lipsindu-i flexibilitatea în fața schimbărilor.

În februarie 2001, șaptesprezece dezvoltatori de software din Statele Unite ale Americii au creat "Manifestul pentru dezvoltarea agilă de software", în speranța de a face un prim pas în depășirea limitărilor pe care le prezentau metodele de dezvoltare a proiectelor software folosite la momentul respectiv. Metodele Agile au cunoscut un adevărat succes începând cu anul 2010. Între anii 2012 și 2015, principiile Agile au devenit majoritare în dezvoltarea software.

Printre metodele Agile dezvoltate până acum, amintesc Scrum, Kanban și Extreme Programming. În ziua de astăzi, cea mai populară metodă Agile este Scrum, folosită în proporție de 56%. Următoarele metode, după procentul utilizării sunt: metode hibride (14%), ScrumBan, un hibrid Scrum-Kanban (8%), hibrid Scrum-Extreme Programming (6%) și Kanban (5%).

Într-un mediu aflat în continuă dezvoltare, importanța metodologiilor Agile a crescut accelerat în ultimii ani. Agile încurajează adaptabilitatea, esențială pentru navigarea schimbării, iar abordarea sa iterativă este o soluție eficientă pentru gestionarea complexității sistemelor. De asemenea, promovează colaborarea, încurajând echipele să depășească provocările împreună. Cu toate acestea, Agile a fost supus criticilor și rezistenței, în special din partea dezvoltatorilor de software. În ciuda originilor sale centrate pe dezvoltator, este uneori privit ca un instrument managerial impus dezvoltatorilor, ceea ce duce la rezistență. Mai mult, comercializarea excesivă a metodologiei Agile poate duce la adoptarea superficială și omiterea principiilor sale de bază.

3. ABORDAREA TEMEI

3.1. TEST-DRIVEN DEVELOPMENT

Test-driven development (TDD) este o abordare Agile care are ca scop creșterea calității produsului final prin testarea codului încă de la început. Pentru ca această practică să funcționeze, programatorii încep dezvoltarea proiectului cu scrierea scenariilor de test, înainte de a începe implementarea propriu-zisă a funcționalității.

Testele proiectate includ criteriile pe care trebuie să le respecte codul, iar programatorii rulează testele fără ca acest cod să fie scris. Testarea este menită să eșueze, astfel încât dezvoltatorii să știe ce trebuie să urmărească în cea de a doua etapă, implementarea funcționalității.

Apoi, dezvoltatorii dezvoltă codul și includ funcționalități care vor face testul să treacă. Odată ce rezultatele testării sunt pozitive, programatorii vor trebui să refactorizeze codul, în vederea obținerii unui cod "curat", păstrând reușitele testelor.

Beneficiile acestei metode includ:

- găsirea erorilor sau greșelilor în primele etape ale implementării
- obținerea codului curat în mod continuu
- dezvoltarea documentației devine mai ușoară, bazându-se pe scenariile de test

Totodată, această metodă poate prezenta și dezavantaje:

- se poate dovedi a fi consumatoare de timp, mai ales în cazul unor dezvoltatori fără experiență
- se poate considera că adaptarea continuă a testelor nu reprezintă o abordare eficientă

3.2. REFACTORIZAREA

Refactorizarea este procesul de rescriere a liniilor de cod pentru a le face mai curate și mai eficiente, fără a modifica comportamentul extern. Refactorizarea este utilă pentru a menține codul ușor de înțeles și modificabil de către alți membri ai echipei.

Practic, refactorizarea implică identificarea și eliminarea oricăror probleme de proiectare, repetiții de cod sau complexitate excesivă care pot afecta calitatea și performanța sistemului. Aceasta poate include reorganizarea structurii de fișiere, extragerea de metode sau funcții, eliminarea codului neutilizat, eliminarea duplicatelor de cod și multe altele.

Refactorizarea nu este doar o îmbunătățire pe termen scurt, ci este un proces continuu în dezvoltarea software. Este adesea efectuată în timpul ciclului de dezvoltare și este considerată o practică esențială în metodologiile Agile, care încurajează echipele să mențină codul curat și să facă îmbunătățiri continue pe parcursul întregului proces de dezvoltare.

Beneficiile acestei metode includ:

- obținerea unui cod curat ușurează munca viitoare a dezvoltatorilor
- îmbunătățirea design-ului sistemului

Totodată, această metodă poate prezenta și dezavantaje:

- încercarea de a refactoriza un cod greu de înțeles sau greșit este consumatoare de timp
- poate conduce la erori în funcționalitate daca nu există destule teste care să atragă atenția asupra omisiunilor

3.3. SIMPLE DESIGN

Simple design este o abordare în dezvoltarea software care promovează crearea de sisteme și aplicații care sunt ușor de înțeles, de întreținut și de extins. Această abordare se concentrează pe păstrarea design-ului cât mai simplu și mai clar posibil, evitând complexitatea inutilă.

Criteriile pe care codul trebuie să le îndeplinească pentru a fi considerat simplu sunt:

- codul este verificat de teste automate, iar fiecare test este rulat cu succes
- codul nu conține duplicate
- codul exprimă intenția programatorilor
- codul este compus dintr-un număr minim de componente, care la rândul lor respectă primele trei criterii

Principiul de simple design contribuie la creșterea calității și a durabilității sistemului. Un design simplu facilitează dezvoltarea continuă și extinderea produsului pe termen lung, oferind o bază solidă și flexibilă pentru adaptarea la cerințele clientului.

3.4. PAIR PROGRAMMING

Pair programming este exact ceea ce sugerează numele: o practică tehnică care permite ca doi programatori să lucreze împreună la dezvoltarea de software.

Pentru ca această tehnică să funcționeze, ambii dezvoltatori trebuie să fie familiarizați cu limbajul de programare și cu proiectul. De asemenea, fiecare dezvoltator trebuie să aibă un rol diferit pentru o perioadă de timp, iar apoi rolurile trebuie schimbate, pentru a reduce greșelile.

Beneficiile acestei metode includ:

- greșelile sunt evitate, deoarece implementarea este făcută de două persoane diferite
- durabilitatea codului este crescută, deoarece există doi dezvoltatori familiarizați cu funcționalitatea și implementarea
- implementarea alături de un partener crește productivitatea și îmbunătățește starea de spirit a celor implicați

Totodată, această metodă poate prezenta și dezavantaje:

- cost crescut, deoarece sunt plătiți doi dezvoltatori, în loc de unul singur
- nu poate fi folosită pe termen lung, nu este posibil ca întregul proiect să fie implementat în acest mod

4. CONCLUZII

În cadrul proiectului, am explorat patru practici tehnice esențiale în metodologia Agile: test-driven development, refactorizarea, simple design și pair programming. Am analizat importanța acestor practici în dezvoltarea software și impactul lor asupra calității, productivității și flexibilității proiectului.

Cu toate că metodologiile Agile pot prezenta și slăbiciuni, iar criticile la adresa lor există, în mod evident, având în vedere domeniul complex care se învârte în jurul aplicării lor, un lucru devine cert: tehnicile practice vin mereu în ajutorul dezvoltatorilor, oferindu-le numeroase posibilități, bine documentate, de abordare a unui proiect software.

Tehnicile practice prezentate în cadrul proiectului reprezintă doar o mică parte din totalitatea tehnicilor dezvoltate sub umbrela metodologiei Agile, ceea ce dovedește faptul că există, cu siguranță, câte o tehnică potrivită pentru orice echipă de dezvoltatori și pentru orice fel de proiect software. Acesta este unul dintre motivele pentru care putem considera că apariția acestei metodologii este extrem de importantă în ceea ce privește dezvoltarea și managementul proiectelor software.

5. BIBLIOGRAFIE

Curs "Managementul Proiectelor Software", Bocan Valer, UPT

https://www.agilealliance.org

https://agilemanifesto.org

"12th Annual State of Agile Report", VersionOne:
https://www.qagile.pl/wp-content/uploads/2018/04/versionone-12th-annual-state-of-agile-report.pdf