Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Departamentul Calculatoare și Tehnologia Informației
Anul universitar 2023-2024

Practici de echipă

Managementul Proiectelor Software

Autor:
ADRIAN POP

Conducător științific:
Ș.l.Dr.Ing. VALER BOCAN

Timișoara
Aprilie, 2024

Contents

1 Introducere

Industria de calculatoare a atins o dezvoltare extrem de rapidă ale cărei amprente se regăsesc peste tot în jur.

Prin comparație cu alte ramuri tehnologice mari, calculatoarele există de cel mai puțin timp, însă oferă cel mai mare impact asupra celorlalte industrii. Dezvoltarea de software reprezintă o bucată semnificativă din multe soluții finite și de aceea, tot mai multă atenție se concentrează asupra părții de proces. În cadrul industriei calculatoarelor, managementul de proiecte software reprezintă o parte vitală din cadrul dezvoltării de produse și servicii. Acesta înglobează practicile folosite pentru a planifica, organiza și controla eficient activitățile și resursele implicate nu doar în dezvoltarea ci și în livrarea unei soluții software.

Asemena dezvoltării oricărui produs, în dezvoltarea de software există o serie de etape care se parcurg de către o echipă de specialiști. În această lucrare se vor discuta teme precum organizarea echipei, crearea și atribuirea rolurilor în cadrul ei, precum și o serie de practici care au rolul de a maximiza rezultatele de care echipa dă dovadă, de a-i spori eficiența și de a exploata cât mai benefic multitudinea de resurse pe care aceasta le pune la dispoziție. Există mai multe soluții standardizate si puse pe piață care vin în ajutorului unui manager de proiect software. Cea mai populară dintre acestea este, pe departe, "Agile" (agil, lb. engleză) care reprezintă mare parte a problematicii aduse în discuție în această lucrare și este, în esență, un set de reguli/principii de urmat.

Multe dintre aceste forme de organizare vin la pachet și se află în strânsă legătură cu diverse unelte ("tools", lb. engleză) pe care cei mai mulți dintre programatori ("developers", lb. engleză) și lideri de echipă le folosesc pentru a-și ușura munca.

O metodă care vine în aplicarea principiilor agile oferind o serie de roluri și obiceiuri ale echipei este "Scrum" - Scrum-Team. Acesta este de asemenea un concept cheie în dezvoltarea lucrării de față.

În mod cert, procesul de dezvoltare software este unul complex. Toate aceste atribute intervin în acest proces pentru a promova colaborarea, calitatea, transparența și adaptabilitatea.

Odată parcursă introducerea în problemă, în al doilea capitol, se vor prezenta, mai pe larg, concepte, studii și rezultate la care s-a ajuns până în acest moment în domeniu. Al treilea capitol va trata în detaliu o serie de practici recomandate într-o echipă de dezvoltare software și particularitățile acestora, aducând și câte o abordare proprie. Ultimul capitol va prezenta concluziile la care s-a ajuns în urma studiului.

2 Analiza Stadiului Actual în Domeniul Problemei

În completarea celor afirmate mai sus, Agile este un set de principii însoțit de valori care, în teorie, ar trebui să îmbunătățească generarea și implementarea ideilor în cadrul procesului creativ. Există mai multe obiective ale dezvoltării software agile și există diferite metodologii pentru abordarea unui Proiect Agile.[Boc24]

În centrul acestui concept se află ideea de a evalua progrsul și de a primi sau oferi *feedback* în mod constant. Figura 1 [Hyg] este o variantă reprezentativă pentru foarte cunoscuta *buclă de feedback* întrucât aceasta ilustrează foarte bine ideea conceptului Agile.

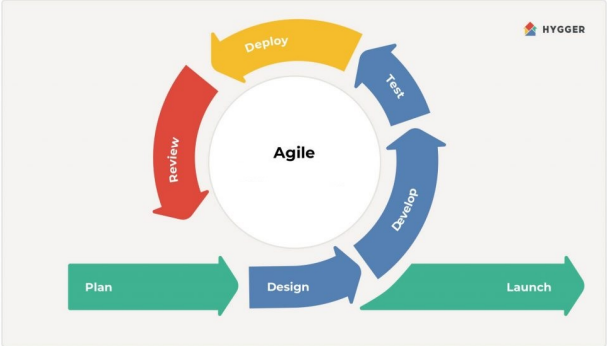

Figure 1: Procesul Agile

Ceea ce este cunoscut ca și "Agile Manifesto" este un document publicat în anul 2001 în SUA și vine cu o serie de 12 principii care, odată implementate promiteau succesul unei echipe. În momentul de față majoritatea echipelor existente aplică, într-o oarecare proporție, acest set de reguli.

Una dintre cele mai populare metodologii care implemetează Manifestul Agile este *Metodologia Scrum*, menționată și ea, anterior. Întrucât, Agile nu este o metodologie, Scrum este un concept ce vine cu o implementare concretă și completă a principiilor. Așa cum se prezintă și în Figura 2 [PM], Scrum definește rolurile în echipă, intervalele orare pentru diferite activități desfășurate de aceasta, dar și "obiectele" cu care membrii lucrează și prin care aceștia interacționează.

Primul rol desemnat într-o echipă Scrum este *Scrum Masterul*. Acesta are responsabilitatea de a coordona activitatea membrilor și de a se asigura că metodele de eficientizare a procesului sunt respectate.

Al doilea rol este *Product Owner-ul*. Acesta se asigură că produsul creat satisafce cerințele clientului și ține practic locul acestuia în echipă.

Niciunul dintre aceștia doi de mai sus nu va participa efectiv la implementarea software-ului în cauză, ci vor coordona împreună activitatea celorlalți participanți.

O serie de ședințe ("meetings", lb. engleză) este definită de procesul Scrum. Denumirile acestora pot varia de la o echipă la alta. De asemenea, metodologia nu prevede orele la care ședințele să se desfășoare, rămânând la latitudnea fiecărei echipe să fixeze aceste detalii atâta timp cât sunt constante.

Prima astfel de sedință este *Daily* sau *Stand-Up Meeting*. În mod evident, aceasta se ține în fiecare zi și este foarte scurtă, de unde și denumirea ce sugerează a fi ținută stându-se în picioare. În timpul întâlnirii moderate de către Scrum Master, fiecare membru are obligația de a oferi un status succint pentru munca sa.

Un "Sprint" (lb. engleză) este o perioadă fixă de timp, de obicei între una și patru săptămâni, în care membrii echipei lucrează pentru a livra o variantă incrementală valoroasă a produsului. Accentul se pune pe versiunea produsului obținută la finalul acestui sprint, scopul principal al său fiind de a crea un produs funcțional și potențial livrabil, care să adauge valoare reală utilizatorilor sau clienților. Astfel, se definesc două alte ședinte care se desfășoara în cadrul sprintului. Întregul proces este astfel împărțit în mai multe iterații cu structură identică.

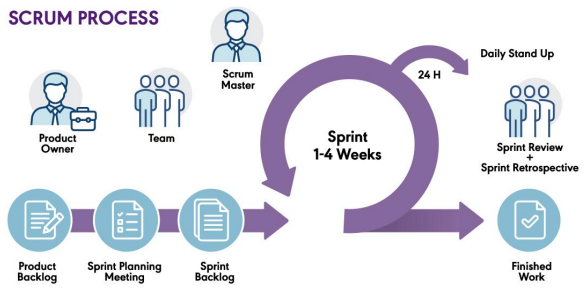

Figure 2: Metodologia Scrum

"Sprint Planning-ul" este ședința de planificare a activității întregului sprint. Sarcinile ("tasks", lb. engleză) care trebuie îndeplinite sunt preluate și împărțite dintr-o listă ordonoată după diferite criterii numită "Product Backlog" din care rezultă ulterior, în urma întâlnirii "Sprint backlog-ul" care nu este nimic altceva decât lista de *tasks* care urmează a fi implementate în interația repectivă.

"Sprint Review/Retrospective" este ședința care încheie sprintul. Odată incheiată iterația, task-urile se reprioritizează și se actualizează *backlog-ul*. În timpul acestei întâlniri, membrii echipei notează și discută punctual ce a mers bine sau ce nu s-a reușit, ce a funcționat cum trebuie și ce nu, dar și chestiuni precum cunoștințe tehnice sau non-tehnice acumulate.

Astfel, se poate afirma că, fiecare iterație este un proces de dezvoltare software în sine, care trece prin toate etapele necesare și se folosește de toate uneltele și resursele disponibile, și căruia i se acordă tot odată, tot interesul, desigur, toate acestea, la o scară redusă.

3 Abordarea Practicilor din Cadrul unei Echipe

Practicile de echipă reprezintă o parte ce determină în mod direct eficacitatea și succesul. Într-o lume în care cerințele se schimbă rapid și tehnologiile evoluează constant, adoptarea unui set solid de practici de echipă devine esențială pentru a asigura livrarea la timp și calitatea a produsului final. Aceste practici abordează diverse aspecte, începând de la managementul timpului și a altor resurse, până la îmbunătățirea colabărării și comunicării într-o echipă. Prin înțelegerea și implementarea acestor practici, echipele de dezvoltare software pot să-și maximizeze potențialul și să-și optimizeze procesele pentru a face față provocărilor dinamice ale industriei. Beneficiile aduse de către fiecare dintre aceste obiceiuri sau forme de organizare sunt distincte și împreună pot creea un mediu de lucru productiv.

"Sustainable Pace" este prima dintre aceste practici și ea s-ar traduce prin sintagma "ritm sustenabil".

Un proiect software nu este o cursă de sprint, ci un maraton [MM06]. Acest citat înglobează ideea ce se află la baza acestei euristici care dorește evitarea fenomenului de "burnout" (epuizare, lb. engleză). Este nevoie să se treacă linia de sosire, o condiție absolut necesară pentru a câtiga, așadar, pentru a ajunge până acolo, toate variabilele trebuie gestionate atent. Nu este în interesul *sportivului* să obțină cel mai bun timp după primii 500 de metri, dacă cursa se încheie după 10km, de exemplu.

Nivelul de stres este monitorizat, iar obiectivele setate fiecărui individ dar și la nivel de echipă trebuie să fie realiste, gestionabile, evitând suprasolicitarea sau subutilizarea resurselor. Este important să se mențină un echilibru între volumul de muncă și capacitatea de a face față acestuia, astfel încât membrii echipei să poată lucra la un nivel optim fără a se confrunta cu epuizarea. Practica *sustainable pace* implică planificarea atentă a sarcinilor și a termenelor limită.

Menținerea unei constanțe în cadrul unei echipe nu este deloc o sarcină ușoară. Această sarcină le este atribuită, bineînțeles, managerilor și ar trbui să fie una prioritară de fiecare dată.

Cel de-al optulea principiu al *Manifestului Agile* punctează și definește un proces de development sustenabil.

"Collective Ownership" (proprietate colectivă, lb. engleză) este o altă practică de echipă extrem de importantă. Esența acestei euristici este ca resposabilitatea și proprietatea asupra întregului proiect să fie distribuită în mod egal între toți membrii echipei.

Așadar, deși poate părea la prima vedere contra-intuitiv, alocarea nu se face pe baza specializării sau a domeniilor specifice de lucru. Această practică încurajează colaborarea și implicarea activă a tuturor membrilor echipei în dezvoltarea și îmbunătățirea continuă a produsului software. Fiecare membru al echipei are dreptul și responsabilitatea de a contribui la orice parte a proiectului, indiferent de domeniul lor de expertiză inițial.

Odată ce toți membri oferă *feedback* și au un grad ridicat de implicare în dezvoltarea proiectului și în implementarea soluțiilor se naște și crește un oarecare devotament pentru produsul final. Printre avantajele care apar pe termen lung în cadrul echipelor care repectă această practică se numără și flexibilitatea sau chiar rezistența la schimbările de personal sau la fluctuațiile de disponibilitate a membrilor echipei. Se elimină prin acest principiu ideea conform căreia unul dintre membri are mai multă autoritate asupra unui modul sau asupra unei tehnologii. Asta nu inseamnă că dacă un individ din echipă are înclinații și aptitudini într-o anumită arie nu va performa majoritar în aceasta, dar va participa și în toate celelate.

"Continuous Integration" (integrare continuă) este una dintre practicile care se dezvoltă într-un mod mai natural deaorace pentru fiecare membru al echipei apare nevoia de a-și integra munca. Există numeroase unelte și programe (*tools*) care vin în ajutorul programatorilor și fac această acțiune mai ușoară. Cele mai impotante și cunoscute sunt Jenkins și Git.

Practica constă într-o integrare automatizată și frecventă a modificărilor efectuate de membrii echipei în codul sursă al proiectului. Scopul principal al integrării continue este de a detecta și rezolva conflictele și erorile într-un stadiu incipient al dezvoltării, astfel încât să se asigure funcționalitatea și stabilitatea codului pe tot parcursul procesului de dezvoltare.

Pe lângă eficiența și economisirea de resurse de timp, această euristică aduce și un plus de comfort pentru membri echipei, deoarece orice membru poate modifica orice modul, oricând [Mar02]. Se ajunge astfel într-un punct în care sistemul suferă în mod automat acțiunea de *build* (construire, lb. engleză) de mai multe ori într-o singură zi, obținându-se astfel de fiecare data o versiune funțională sau nu a produsului dezvoltat.

Când vine vorba despre **"Sprint Meetings"**, durata acestora este foarte importantă. *Sprint Planning-ul* și *Sprint Retrospective/Review* sunt ședințele de echipă care apar într-un mod mai natural. Planificarea *task-urilor* de lucru trebuie realizată în orice caz și, la un moment dat, trebuie catalogate o serie de rezultate sau implemetări ca fiind satisafăcătoare sau nu. În funcție de numărul de membrii sau de complexitatea proiectului, acestea două nu ar trebui să deășească 2 ore.

În cazul *Daily-ului/Stand Up-ului* durata este și mai importantă. Practica constă într-o ședință fixată zi de zi în care fiecare membru exprimă punctual ceea ce a realizat cu o zi înainte, ce are de gând să realizeze în ziua curentă și în care să menționeze dacă a vut blocaje sau nu. Toate acestea rămân însă la stadiul de mențiune. În timpul *meeting-ului* nu se caută și nu se dezbat soluții, astfel că durata de desfășurare recomandată este de doar 15 min. Se poate spune, din acest punct de vedere, că întreaga ședință este una de constatare.

Scopul este acela de a împărți munca echitabil și sănătos de-a lungul sprint-ului. Dacă fiecare membru dobândește la finalul zilei un mic progres, echipa denotă, la final de sprint, progres.

O altă practică de echipă des întâlnită în software development este **"Pair Programming"**. Aceasta presupune ca doi software developers să scrie codul în fața aceluiași monitor, ca și în figura 4, interacționând continuu. Atribuția de a tasta efectiv codul ar trebui des interschimbată, iar un developer ar trebui să facă parte din minim două astfel de perechi în timpul zilei de lucru. Această practică crește exponențial volumul de cunoștnțe dobândit în cadrul echipei.

Deși ar putea părea inițial că practica scade eficiența echipei, numerease studii au demostrat contrariul. Ceea ce scade, însă, este apariția defectelor (*"bugs"*, lb. engleză), grație interacțiunii intense dintre coechipieri.

"Test Driven Development" (TDD) este o altă euristică de echipă, de această dată mai puțin îndrăgită de programatori și mai puțin implementată de către manageri. Codul este scris, în cadrul acestei tehnici, în paralel cu testele pentru funcționalitatea acestuia. Procesul este ghidat de teste, ceea ce înseamnă că înainte de a scrie codul propriu-zis pentru o anumită funcționalitate, dezvoltatorul începe prin a scrie un test care definește comportamentul dorit al acelei funcționalități. Apoi, dezvoltatorul completează codul necesar pentru a face testul să treacă cu succes. Pot exista chiar și cazuri extreme, în care toate testele să fie scrise înaintea începerii implemetării.

Avantajele practicii sunt clare : detectarea timpurie a erorilor, scrierea codului mai curat și mai modular, creșterea încrederii în cod și reducerea timpului necesar pentru depanarea și remedierea bug-urilor. Cu toate acestea întreaga abordare, organizare și structură a proiectului se schimbă în cazul deciziei de a implemta această tehnică.

Figura 3[Ech] de mai jos, prezintă o statistică a metodologiilor folosite în *software development* pentru anul 2023.

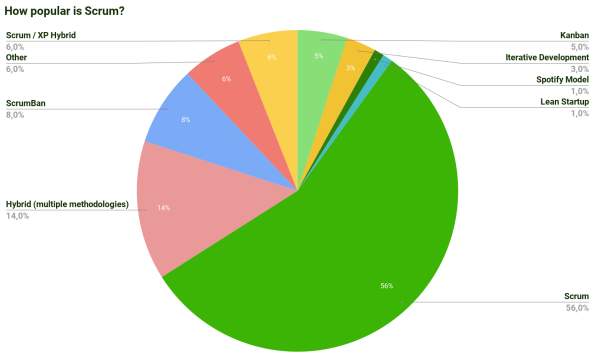

Figure 3: Procent metodologii 2023

4 Concluzii

Managementul de proiecte software și practicile de echipă asociate reprezintă elemente fundamentale pentru succesul în dezvoltarea și livrarea de produse software de calitate.

Studiul realizat demostrează prin dezbaterea practicilor de echipă existente că deși țelurile ce se doresc a fi atinse sunt calitatea software-ului obținut, eficiența în timp sau chiar reducerea costurilor totale, cele mai impactante măsuri sunt luate în direcții de tip *"people centered"* (orientate spre oameni, lb. engleză) și nu *"task/result centered"* (orientate spre sarcină/rezultat, lb. engleză).

Prin analiza stadiului actual în domeniul managementului de proiecte software, am observat că metodologiile Agile, cum ar fi Scrum, sunt extrem de populare și aduc beneficii semnificative în ceea ce privește flexibilitatea, adaptabilitatea și livrarea incrementală a produselor. Principii cum ar fi *feedback-ul* constant, colaborarea între membrii echipei și adaptabilitatea la schimbări, sunt esențiale pentru succesul unui proiect software într-un mediu volatil și imprevizibil.

Este totuși esențial să afirmăm că nu există o soluție unică sau o abordare universal valabilă în managementul de proiecte software. Practicile preznetate în această lucrare constituie un ghid cel puțin folositor pentru îmbunătățirea proceselor și rezultatelor în dezvoltarea de software, însă, fiecare echipă are nevoi și contexte specifice. Cel mai important este felul în care se adoptă și adaptează aceste practici în cadrul fiecărei echipe.

Figure 4: Interacțiunea în cadrul echipei [Bug]

References

[Boc24] Valer Bocan. *Curs MPS: Managementul Proiectelor Software*. 2024.

[Bug] Shake Bugs. Software team collaboration: Best practices. Accesat Aprilie 2024.

[Ech] Echometer. The 20+ most important scrum statistics for 2023. Accesat Aprilie 2024.

[Hyg] Hygger. Agile guide. Accesat Aprilie 2024.

[Mar02] Robert C. Martin. *Agile Software Development: Principles, Patterns, and Practices*. 1st edition, 2002.

[MM06] Micah Martin and Robert C. Martin. *Agile Principles, Patterns, and Practices in C#*. 1st edition, 2006.

[PM] PM Partners. The agile journey: A scrum overview. Accesat Aprilie 2024.