Tema : Modele de dezvoltare software suplă, agilă(introducere + partea 1/3)
Student : Caraivan George-Alexandru 441A

Modele de dezvoltare software suplă, agilă

Introducere

Modele de dezvoltare agilă sunt un set de metode iterative și incrementale de dezvoltare a produsului soft. Cele mai renumite dintre acestea sunt programarea extremă, scrum, DSDM, FDD. Deși fiecare dintre aceste metode este unică prin abordare ele au în comun valorile și viziunile descrise de manifestul agile[2]. Toate aceste metode implică comunicare, planificare, testate și integrare continuă care ajută la dezvoltarea unui soft bun dar ceea ce definește o metodă agilă este faptul că ele încurajează colaborarea dintre oameni și face ca deciziile luate în comun să fie rapide și bune.

Fiecare dintre metodele agile este de fapt o soluție ce incorporează anumite practici și principii asociate diferitelor părți componente ale unui proiect și are ca scop ghidarea echipelor astfel încât procesul de dezvoltare să se desfășoare cât mai rapid și mai eficient.

Deși aceste practici ale dezvoltări agile au apărut de ceva vreme ele încă nu au fost abordate pe scară largă de echipe soft dar odată cu dezvoltarea modelelor adică combinarea lor într-o modalitate care să fie ușor de înțeles și de implementat acest lucru pare să se schimbe în bine.

Procesul agil

Procesul agil se referă în principal la crearea unui cadru care să permită unei echipe să fie agilă[5]. Realizarea felului în care o echipă poate lucra ca un întreg este mult mai important decât orice alt proces deoarece actul de a permite echipei să lucreze ca un întreg are un efect mult mai benefic asupra productivitații decât un nou proces care poate îmbunătăți productivitatea echipei doar cu mică fracțiune.

Includerea clientului în echipă este schimbarea cea mai importantă a procesului. Prezența lui asigură faptul că fiecare poate învăța ceva nou despre problema de rezolvat și în plus experiența lui în domeniu facilitează găsirea unei soluții simple, elegante și bineînțeles corecte.

Procesul agil acceptă realitatea schimbărilor cerințelor iar proiectele care accepta acest lucru sunt mult mai eficiente din punctul de vedere al costului față de proiectele care nu acceptă

că cerințele se pot schimba. În locul gestiunii activităților procesul agile gestionează cerințele iar acest lucru deschide noi posibilități de dezvoltare.

Fiecare trăsătură a proiectului dorită de client este reprezentată ca o poveste scrisă pe o singură foaie. Cu ajutorul acestor povești se poate estima întinderea proiectului și vizualizarea activităților pentru ca mai apoi să se creeze un calendar în care trăsăturile vor intra în ordinea importanței. În acest mod se poate obține o estimare a costului unei activități și se poate decide asupra ceea ce poate fi lăsat nefăcut într-un mod mai inteligent.

Procesul agil se bazează pe faptul că un design mai simplu și mai elegant al proiectului este mult mai valoros decât unul complex și greu de întreținut.

Programarea extremă

Programarea extremă este cea mai folosită dintre metodele agile. Deși împărtășește valorile exprimate în manifestul agile ea caută acele practicile simple și absolut necesare care pot rezolva problema. Valorile de bază ale programării extreme sunt: simplitatea, comunicarea, feedback și curajul[10]. Prin acestea o echipă poate deveni conștientă de sine și poate estima în mod sincer progresul și își poate rafina practicile astfel încât să se adapteze la situația prezentă.

O echipă este alcătuită astfel încât să poată aborda toate aspectele dezvoltări și are în centrul unul sau mai mulți reprezentanți ai clientului cu care lucrează zi de zi.

Echipele folosesc o formă de planificare simplă pentru a decide ce urmează să fie făcut și pentru a ști când și ce se poate livra. Produsul software este alcătuit dintr-o serie de livrări la scară mică dar care sunt total integrate și trec toate testele definite de client. Pentru lucrurile definite mai sus practicile programării extreme poartă numele de echipa întreagă, jocul planificării, livrări mici, teste de acceptare.

Programatorii unei echipe lucrează împreună în echipe de 2 sau ca grup și au în vedere un design simplu mereu adaptabil la cerințe ce are la bază un cod bine testat. Practicile poartă numele de programarea în perechi, designul simplu, dezvoltarea bazată pe testare, îmbunătățirea design-ului.

Echipa menține sistemul integrat și are grijă ca el să ruleze tot timpul iar programatorii codează într-un mod consistent astfel încât fiecare să poată înțelege sau chiar îmbunătățți codul conform cu cerințele care apar. Practicile poartă numele de integrare continuă, codare standard și codul aparține echipei.

Ca întreg echipa trebuie să împărtășească o image simplă a felului în care sistemul arată și fiecare trebuie să lucreze cu o viteză care să poate fi menținută în mod nedefinit. Practicile poartă numele de metaforă și viteza susținută.

Ansamblul de practici poarta numele de Circul Vieții[12] și este reprezentat ca în figura următoare:

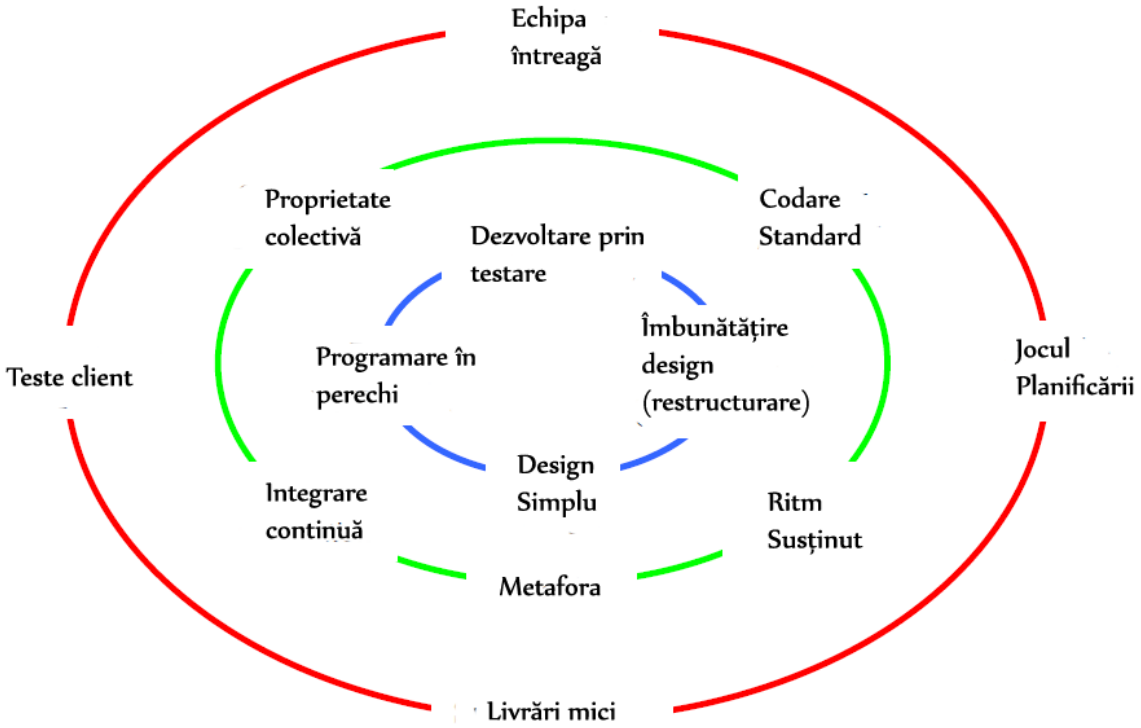

Figura 1: Practicile programării extreme[12]

Strategii XP de management, planificare, design, testare

Strategia de management poate fi ușor exprimată prin termenii : livrarea în etape, feedback concret și rapid, exprimarea articulată a cerințelor sistemului și specialiști pentru sarcine speciale[11]. Dilema managementului este exprimată prin faptul că dorim ca cineva să ia toate deciziile dar acest lucru este imposibil deoarece nu există o singură persoană care să știe atât de multe lucru astfel încât să poată să ia toate deciziile. Cu

toate acestea trebuie să existe cineva care să aibă o viziune amplă și care să poată să țină proiectul pe calea sa naturală.

Unealta de bază în management este folosirea metricilor. Un exemplu ar putea fi raportul dintre timpul estimat pentru dezvoltare și timpul calendaristic cu ajutorul căruia se poate estima dacă procesul se desfășoară în parametrii stabiliți sau dimpotrivă. Deoarece este dificil ca o echipă să urmărească mai multe metrici la un moment de timp, unele metrici sunt scoase după ce și-au servit scopul iar pentru eficiență este bine ca numărul lor să nu depășească 4 într-un moment dat. Scopul metricilor este acela de comunica nevoia de schimbare atunci când acest lucru este necesar.

O altă metodă de management este folosirea unei persoane menită ai ajuta pe fiecare să facă deciziile corecte iar sarcinile sale sunt de a fi un partener în dezvoltare pentru sarcinile mai dificile, de a ajuta programatorii cu abilitățile tehnice precum testarea, restructurarea și nu în ultimul rând pentru a explica procesul managerilor (de la nivelul de sus).

Măsurarea efectivă a ceea ce se întâmplă și nu doar estimarea este o altă componentă a managementului. Cel care face măsurătorile trebuie să comunice echipei ceea ce s-a făcut și să amintească ceea ce se dorește să se facă sau ceea s-a presupus că se va face. Colectarea datelor se face de obicei de 2 ori pe săptămână.

Cea mai dificilă problemă a managementului este intervenția. Acest lucru este caracterizat de acele momente în care managerul trebuie să ia măsuri prin decizii dificile cum ar fi schimbarea de personal, sau decizia de a opri proiectul și a începe altul.

Planificarea este procesul în care se imaginează cum ar trebui să fie dezvoltarea unui produs software. Prin acest proces se urmărește formarea echipei, stabilirea priorităților, estimarea costului și crearea agendei, stabilirea procesului de feedback și nu în ultimul rând găsirea încrederii că sistemul poate fi realizat.

Procesul este abstractizat sub forma unui joc între 2 participanți : dezvoltatorul și afaceristul care sunt nevoiți să joace după niște reguli impuse. Scopul jocului este maximizarea valorii softului produs de echipă, din aceasta se deduce costul dezvoltării și riscurile inerente. Strategia este ca echipa să investească cât mai puțin pentru ca produsul să aibă funcționalitatea dorită cât mai rapid.

Dezvoltatorul este reprezentat de toți acei oameni care vor fi responsabili de implementarea sistemului iar afaceristul de toți acei oameni care decid ce trebuie să facă sistemul.

Jocul are 3 faze : explorarea, dedicarea și îndrumarea.

Faza de explorarea are ca scop formarea unei imagini de asamblu și implică descrierea a ceea ce sistemul ar trebui să facă, stabilirea timpul necesar pentru a implementa anumite părți și prioritizarea.

Faza de dedicare își propune să stabilească data următoarei livrări și să se asigure că acest lucru se va întâmpla. Afaceristul stabilește ce este absolut necesar pentru ca sistemul să funcționeze, ce este mai puțin necesar dar poate oferi valoare economică și ce facilități s-ar dori să fie implementate. Dezvoltatorul decide ce poate implementa cu o estimare precisă, rezonabilă sau ce nu se poate estima iar apoi comunică afaceristului cât de rapid poate lucra echipa într-o lună iar pe bază acestei informații se va stabili întinderea proiectului.

Faza de orientare ajută în ajustarea planului pe măsură ce apar date noi. Se aleg iterațiile cele mai importante, prima iterație fiind una în urma căreia trebuie să rezulte un sistem funcțional oricât de incipient ar fi. Putem avea diferite scenarii: dacă viteza a fost supraestimată dezvoltatorul comunică afaceristului ce poate implementa pentru versiunea curentă având ca bază noua viteză de dezvoltare, dacă afaceristul realizează că trebuie implementat ceva nou, dezvoltatorul va estima cât va dura și afaceristul va fi nevoit să șteargă implemetări care au un estimat echivalent pentru a introduce noua implemetare, dacă dezvoltatorul decide că planul nu mai este de acuratețe, poate reface estimarea pentru toate implemetările ce urmează a fi integrate și setează noua viteză a proiectului.

Programarea extremă cere design**-ului** să fie cel mai simplu design care trece de suita de teste curentă. Folosind valorile programării extreme trebuiesc create : o strategie de design care să rezulte într-un design simplu, o metodă de a verifica calitatea design-ului și manipularea ciclului de timp astfel încât acest proces să fie cât mai scurt posibil.

O strategie este adăugarea a ceea ce este nevoie acum sau mai târziu dar problema acesteia este incertitudinea. Există posibilitatea inexistenței unui timp mai târziu sau se pot găsii căi prin care cei doi timpi se pot fi interconectați. Din acest motiv este de dorit ca viziunea să fie schimbată astfel încât să

reflecte ce design se va face azi pentru problemele de azi și ce design se va face mâine pentru problemele de mâine.

Strategia de design începe prin crearea unui test astfel încât să se știe momentul final și cu ajutorul căruia se va implementa o parte din design. După aceea se implementează designul astfel încât să poată fi trecut acel test și va rezulta un design suficient al implementării ca să trecem testul și toate cele dinaintea lui. Etapa a treia este repetarea strategiei începând cu primului pas cu mențiunea că dacă se întrezărește posibilitatea de a face design mai simplu se va executa acest lucru. Deși pare simplă această strategie este capabilă să creeze sisteme sofisticate.

Simplitatea designului este măsurată prin 4 constrângeri prioritare : sistemul trebuie să comunice tot ceea ce se dorește ca acesta să comunice, lipsa codului duplicat și un număr cât mai restrâns de clase iar apoi de metode[11].

Testele în programarea extremă sunt izolate prin faptul că un test nu interacționează cu celelalte teste și ele trebuie să fie automate.

Deoarece este imposibil de făcut teste pentru orice lucru ele ar trebui făcute doar pentru acele lucruri care se pot defecta. Testele sunt scrise de programatori și de clienți, diferența constând în faptul că programatorii scriu teste bazate pe metode pe când clienții scriu teste bazate pe scenarii.

Deși testele de unitate și funcționalitate stau la baza strategiei de testare în programarea extremă, din când în când se folosesc și altfel de teste precum teste de stres care simulează cele mai dificile cazuri și teste paralele care arată în ce fel sistemul nou diferă de cel vechi.

Implementare XP și cicluri de viață

Cea mai dificilă parte a programării extreme este implementarea, implicit acomodarea. Algoritmul pentru a adopta programarea extremă este simplu : alege cea mai grea problemă, rezolv-o folosind strategia programării extreme iar atunci când nu mai este cea mai grea problemă alege alta. Cel mai facil loc de plecare este jocul planificării. De asemenea în adoptarea programării extreme nu trebuie subevaluată importanța locului fizic care trebuie să rearanjat astfel încât programarea să se poată face în perechi și clientul să poată fi pe aproape.

Un proiect de programare extremă ideal trece inițial printr-un stagiu foarte scurt de dezvoltare urmat apoi de ani de suport și rafinare pentru ca în final când proiectul nu își mai are rostul să se înceapă retragerea.

Programare expremă se bazează pe ideea că stările în care nu se produce nimic sunt stări nenaturale. Totuși pentru a putea trece în starea de producție este necesară starea de explorare în care se afirmă posibilitatea realizării proiectului și se poate întrezări o primă versiune de livrare care să fie suficient de bună. În această fază se explorează posibilitățile pentru arhitectura sistemului și se experimentează cu limitările de performanță ale tehnologiilor dorite a fi folosite, de asemenea se estimează fiecare sarcină. Pentru o echipă în care membrii deja se cunosc și își cunosc și tehnologiile, faza de explorare se poate limita la câteva săptămâni, în schimb pentru o echipă relativ nouă acest lucru se poate întinde pe parcursul câtorva luni.

Scopul planificării este ca programatorii și clienții să cadă de comun asupra unui moment în timp în care implementările cele mai importante vor fi finalizate. Cu o pregătire bună în etapa de explorare procesul de planificare poate dura chiar 2 zile.

Fiecare iterație dinainte de prima lansare are rolul de a produce un set de teste de funcționalitate pentru fiecare dintre implementările stabilite a fi incluse în acea iterație. Prima iterație are rolul de a crea scheletul arhitectural iar apoi cu fiecare iterație se mai adaugă implementari având grijă la deviații. Detectarea lor înseamnă că planul trebuie schimbat sau poate chiar procesul. După ultima iterație proiectul ar trebui să fie pregătit să intre în producție.

În etapa de producție se realizează o modificare a iterațiilor de la o durată de 2-3 săptămâni la o durată de o săptămână și se fac întâlniri de maxim 15 minute în fiecare zi pentru ca fiecare membru al echipei să știe asupra ce lucrează ceilalți.

În această etapă apare procesul de certificare și de îmbunătățire a performanței iar în acest moment ar trebui încetinit ritmul de evoluție al proiectului deoarece riscul devine din ce în ce mai important în evaluarea a ceea ce trebuie modificat. Când proiectul chiar ajunge în producție echipa ar trebui să dea o petrecere să celebreze faptul că proiectul este în producție și să se folosească de ea pentru a elimina stresul[11].

Faza de mentenanță este cu adevărat starea normală a unui proiect de programare extremă și în care se produc noi funcționalități simultan cu pastrarea sistemul activ, incorporarea de noi membri

în echipă sau despărțirea de alții care se vor orienta spre proiecte noi. Dezvoltarea unui sistem în etapa de producție este diferită de dezvoltarea unuia care nu este în această etapă deoarece trebuie să fii mai atent la schimbări și să fii pregătit pentru întreruperea dezvoltării pentru a reacționa la problemele care apar în producție.

Atunci când clientul este mulțumit cu sistemul și nu mai găsește noi idei de implementat atunci este timpul ca proiectul să ia sfârșit. În această ultimă etapă se scrie un document de 5-10 pagini care va reprezenta un ghid al sistemului pentru eventualitatea în care cineva va dorii să mai schimbe ceva cândva.

Pe baza conceptelor de mai sus ritmul procesului programării extreme poate fi reprezentat ca în figura următoare :

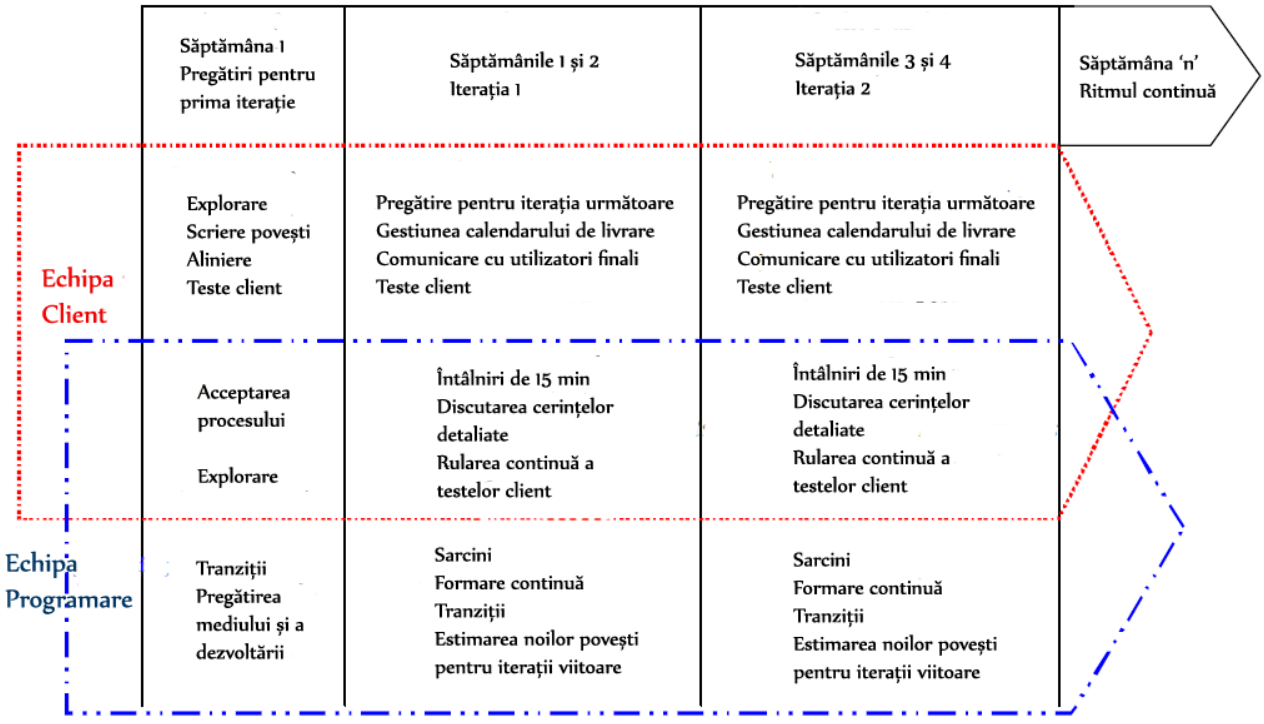

Figura 2 : Ritmul programării extreme[12]

Oameni și roluri în XP

Programatorii stau la baza XP și ar putea face decizii care să balanseze prioritățile pe termen lung cu cele pe termen scurt atunci proiectul ar avea tot personalul necesar.

Sarcina programatorului nu se termină când calculatorul înțelege ce trebuie să facă ci când toate componentele comunicației cu ceilalți au fost realizate și pe baza acestora se pot crea teste care să demonstreze aspectele vitale ale sistemului.

Obiectivul programatorului este să dezvolte un soft cât mai valoros pentru client în timp ce evită dezvoltarea oricărui lucru

lipsit de valoare. Printre abilitățile cerute pentru un programator se numără abilitatea de a lucra în echipe de 2 oameni, simplitatea reflectată în discuțiile cu clientul dar și în codul scris.

Abilitățile tehnice implică pe lângă faptul de a fi un programator bun și capacitatea de a testa codul, de al îmbunătății și de putea renunța la posesivitatea asupra codului personal prin încrederea în modificările făcute de ceilalți și prin posibilitatea de a învăța din ele. Nu în ultimul rând trebuie să aibă curajul să își înfrunte frica de a părea incapabil, nefolositor sau nu suficient de bun.

Clientul este cealaltă dualitate necesară programării extreme. În timp ce programatorul știe cum să programeze, clientul știe ce trebuie să fie programat. Să fii un client nu este ușor deoarece trebuie să știi să descrii ce implementări vrei și să ai o atitudine de câștigător. Cea mai grea parte este luarea de decizii și păstrarea confidenței atunci când lucrurile nu merg așa cum ar trebui.

Rolul testerului este de a fi axat pe client și de al ajuta să scrie și să aleagă testele de funcționare. Testerul nu este o persoană separată dedicată în mod exclusiv umilirii programatorilor ci este cineva care are responsabilitatea de a rula teste în mod regulat, de a anunța rezultatele și de a se asigura că uneltele de testare merg bine.

Persoana care monitorizează sistemul este supranumită conștiința echipei. Ea trebuie să facă estimări bune și vadă dacă ele se conformează realității. Sarcina ei este de a inchide bucla de feedback și de păstra o viziune de ansamblu. De asemenea are responsabilitatea de a păstra rezultatele la testele de funcționare și abilitatea cea mai importantă este aceea de a colecta informația necesară fără a deranja prea mult procesul.

Antrenorul este cel care este răspunzător pentru întregul proces și care observă când apar deviații și comunică acest lucru echipei. El este cel care are responsabilitatea de a ghida echipa prin sugestii și observarea erorilor, lucruri ce implică siguranță de sine și o bună pregătire necesară controlării situațiilor conflictuale ce pot apărea.

Consultantul este ajutorul echipei atunci când ea necesită cunoștințe tehnice profunde. Rolul lui este de a ajuta echipa în astfel de probleme și de ai învăța cum să rezolve singuri această problemă.

Seful este omul care trebuie să aibă curaj, confidență și insistență. Este cel care trebuie să aibă grijă de bunul mers al echipei și care trebuie să intervină doar atunci când echipa are cu adevărat probleme pe care doar el le poate rezolva. În majoritatea timpului trebuie să stea cât mai departe dar atunci când abilitățile sale sunt cerute el trebuie să își ajute echipa în mod absolut fie că e vorba despre nevoia unui nou tester sau acceptarea unor idei foarte îndrăznețe.

Critici XP

Tranziția către programarea extremă este foarte grea și ideea de a face lucruri simple nu este foarte simplă. De asemenea este foarte greu să admiți că nu știi, este foarte greu să colaborezi cu ceilalți, este greu să spargi zidurile emoționale.

Se consideră că ea funcționează doar cu dezvoltatori seniori și că îi lipsește structura și documentația necesară. În anumite cazuri poate fi foarte ineficientă și poate fi imposibil de estimat efortul lucrativ într-un mod realist. Deoarece se bazează pe trăsături, atributele de calitate nefuncționale pot fi greu descrise ca povești ale utilizatorul iar în final metodologia este atât de efectivă pe cât de efectivi sunt oameni implicați lucru nerezolvat de mentalitatea agilă.

Concluzii

Metodele agile pot fi ineficiente pentru anumite tipuri de proiecte sau pentru organizații extinse. De asemenea multe organizații consideră că metodele sunt prea extreme și de aceea ele adoptă niște metode hibride[1]. Cu toate aceastea metodele agile sunt considerate bune pentru proiectele evolutive și nesecvențiale[1].

Deși nu există o metodologie care să fie bună pentru orice tip de proiect se poate observa că totuși programarea extremă oferă un set de idei și practici testate, documentate și care pot fi intregrate în metodologia folosită pentru dezvoltarea eficientă a proiectului.

Bibliografie

[1] http://en.wikipedia.org/wiki/Agile_software_development

[2] http://agilemanifesto.org/principles.html

[3] http://net.tutsplus.com/articles/general/the-principles-of-agile-development/

[4] http://www.versionone.com/Agile101/Agile-Development-Overview/

[5] http://www.agile-process.org/

[6] http://blogs.hbr.org/cs/2012/10/how_we_finally_made_agile_development_work .html

[7] http://en.wikipedia.org/wiki/Extreme_programming

[8] http://xprogramming.com/book/whatisxp/

[9] Introduction to Extreme Programming by Mike Grant

[10] Extreme programming by Ganesh Sambasivam

[11] Extreme programming explained by Kent Beck

[12] Extreme programming and agile software development methodologies by Lowell Lindstrom and Ron Jeffries