METODE *AGILE* DE MANAGEMENT AL PROIECTELOR

lect. dr. Dan Mircea SUCIU

Cluj Napoca

2018

Metodologii Agile de Management al Proiectelor

Dan Mircea Suciu

This book is for sale at http://leanpub.com/Metode_Agile

This version was published on 2018-08-30

ISBN 978-973-0-27742-5

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Cuprins

Introducere....6
1. Managementul cunoștințelor. Particularitățile proiectelor *software* 9
Analiza succesului proiectelor software....9
Caracteristicile proiectelor software....12
Softul este nemăsurabil....12
Softul este invizibil și intangibil....14
Softul are o complexitate ridicată....14
Softul este dinamic....15
2. Filozofia Agile....17
Manifestul Agile....17
Principiile Agile....18
Management tradițional și management Agile....20
3. Evoluția metodologiilor Agile....22
Sistemul de clasificare a problemelor *Cynefin*....22
Evoluția metodologiilor Agile....26
Metodologii Agile relevante....28
4. *Scrum*: roluri, evenimente și artefacte....31
Istoric....31
Roluri....32
Evenimente....33
Instrumente (Artefacte)....34
5. *Extreme Programming*: valori, principii și practici....36
Valori XP....37
Comunicarea....37

Simplitatea....37
Feedback-ul....38
Curajul....38
Respectul....39
Principii....39
Feedback rapid....39
Asumarea simplității....40
Modificare incrementală....40
Îmbrățișarea schimbării....41
Muncă de calitate....41
Practici XP....41
6. Lean Software Development....45
Principii *Lean*....46
Principii *Lean* pentru dezvoltarea de software....47
7. Kanban....50
Diagrame de flux cumulativ....51
Implementări Kanban în proiecte IT....53
Implementare Kanban....55
8. Alte metodologii Agile....57
Feature Driven Development (FDD)....57
Agile Unified Process (AUP)....59
Crystal....61
Dynamic Systems Development Method (DSDM)....62
Concluzii....63
9. Contracte Agile. Gestionarea riscurilor....65
Contracte Agile....65
Contract DSDM....66
Contracte tip *Money for Nothing* și *Changes for free*....66
Contracte cu prețuri graduale fixe....67
Contracte cu preț fix per pachet....68
Gestionarea riscurilor în proiecte Agile....68

10. Piramida lui Dilts și adaptarea la schimbare 71
Nivele logice de învățare și... schimbare 71
Capcane în adoptarea Agile.....75
Concluzii 76
11. Mentalitate de produs (*Product Mindset*).....78
Companii orientate pe Servicii (CoS) și Companii orientate pe Produs (CoP).78
Product Mindset 81
Concluzii 83
12. #NoAgile (în loc de concluzii) 84
#NoEstimates 85
#NoProjects 87
#NoBacklog.....88
#NoSprints/#NoReleases.....89
Concluzii 90
Bibliografie 92

Introducere

Cartea de față își propune să sintetizeze în paginile sale caracteristicile celor mai populare metodologii și practici utilizate în managementul proiectelor într-o manieră agilă. Primele trei capitole tratează acest subiect în ansamblu său, explicând necesitatea acestor metodologii în contextul actual, domeniile în care ele pot fi utilizate cu succes și evoluția lor de-a lungul timpului.

Următoarele capitole se concentrează pe descrierea unor metodologii și practici particulare, începând cu *Scrum* și *Extreme Programming* și continuând cu abordările *Lean* (insistând pe *Kanban*) și cele derivate din metodologiile de management predictiv cum sunt *Feature Driven Development* și *Agile Unified Process*. De asemenea, sunt amintite și alte două metodologii ce au fost utilizate pe o scară restrânsă în managementul proiectelor (*Crystal* și *Dynamic Systems Development Method*) dar care au avut o mare importanță în întreaga istorie a abordărilor Agile, în special în facilitarea tranziției de la predictiv la agil.

Capitolele finale discută capacitatea echipelor de proiect de a adopta metodologiile Agile și sunt analizate mai multe curente ce vor influența abordările Agile în viitor.

Înainte de toate, această carte se adresează celor care doresc să se familiarizeze cu domeniul managementului Agile al proiectelor. Prin urmare este utilă celor care doresc să înțeleagă care sunt avantajele și riscurile utilizării practicilor Agile, care sunt tipurile de proiecte în care acestea se recomandă să fie utilizate și ce alternative sunt disponibile.

În același timp, cartea este utilă și celor care au o experiență mai vastă în implementarea unor metodologii particulare în echipele lor de proiect însă care se confruntă periodic cu diverse provocări în ceea ce privește organizarea echipei, implicarea clientului sau respectarea anumitor constrângeri ce influențează hotărâtor succesul proiectului.

Cartea de față nu este însă recomandată ca suport principal în implementarea efectivă a unei metodologii particulare într-o echipă de proiect. Acest lucru ar fi presupus adăugarea unor detalii ce ar fi crescut complexitatea conținutului, lucru care ne-ar fi abătut de la scopul declarat inițial, și anume sinteza.

Nu în ultimul rând, merită adăugat aici faptul că această carte acoperă o lipsă cu care se confruntă astăzi literatura românească de specialitate. Subiectul metodologiilor de management Agile se regăsește în foarte puține cărți apărute în limba română (ca variantă originală sau tradusă), iar aria de răspândire a acestora este redusă. Este adevărat, majoritatea celor interesați de acest subiect își desfășoară activitatea în domenii în care, în general, limba engleză este foarte bine cunoscută, iar oferta de lucrări pe subiectul Agile este în acest caz foarte generoasă. Acest lucru nu justifică însă absența tratării subiectului aproape complet în limba română. Acesta este și principalul motiv pentru care a fost destul de dificil de găsit, în anumite momente, variantele românești ale unor termeni care, în echipele de proiect curente sunt utilizați exclusiv în limba engleză. De aceea se regăsesc în numeroase locuri pe parcursuri cărții, alături de termenii în limba română și corespondentul acestora în engleza, cum ar fi *backlog*, *user story*, *cycle time*, *release* sau chiar *best practice*.

Am construit această carte avându-i constant în minte pe studenții mei. În ultimii 3 ani am predat cursul de *Metodologii Agile de Dezvoltare a Softului* la patru secții de master al Facultății de Matematică și Informatică a Universității Babeș-Bolyai din Cluj Napoca și cursul de *Metode Alternative de Management al Proiectului* la grupa de master în Management a Școlii Nationale de Studii Politice și Administrative din București. O bună parte din capitolele acestei cărți respectă structura implementată în aceste două cursuri. De asemenea, foarte multe dintre comentariile și observațiile primite de la studenți și-au găsit locul în diverse forme în această carte, motiv pentru care le mulțumesc din suflet.

Patru dintre capitole (dispuse de început și final își au originile în tot atâtea articole publicate în revista lunară de IT *Today Software Magazin*, ele fiind adaptate și încadrate în contextul cărții. Îi mulțumesc fondatorului revistei, Ovidiu Mățan, pentru perseverența cu care m-a încurajat să le scriu și apoi să le prezint în diverse evenimente de lansare ale revistei în comunitatea IT clujeană și nu numai.

Această carte nu ar fi existat fără discuțiile intense și provocatoare pe tematici Agile și conexe avute cu Simona Bonghez și Bogdan Mureșan de-a lungul timpului și cărora le mulțumesc că mi-au relevat perspective cu totul aparte asupra acestor tematici. Cele mai interesante idei pe care le-am dezbătut împreună se află răsfirate printre rândurile acestei cărți.

Un gând final merge către familia fără de care nu aș fi fost capabil să învăț, să mă dezvăț și să re-învăț atât de multe lucruri. Mulțumesc!

1. Managementul cunoștințelor. Particularitățile proiectelor *software*

Metodologiile Agile de gestionare a proiectelor sunt utilizate într-o categorie aparte de proiecte, și anume în cele în care majoritatea activităților de proiect se bazează pe prelucrarea cunoștințelor membrilor echipei de proiect, a organizației din care fac parte aceștia dar și a celorlalte părți implicate în proiect (*stakeholders*).

Încă din anul 1950 Peter Drucker a introdus conceptul de "*knowledge workers*" pentru caracterizarea muncitorilor capabili să utilizeze cunoștințele unei organizații la realizarea unor produse intangibile [1]. Astăzi o categorie însemnată de proiecte se bazează pe resursa critică *cunoștința*. Alternativele de gestionare a acestor proiecte sunt orientate spre găsirea unor principii, metode și tehnici de analiză, planificare, organizare și transfer a cunoștințelor, toate acestea regăsindu-se astăzi sub denumirea generică de management al cunoștințelor.

Cea mai relevantă categorie de proiecte ce se bazează aproape exclusiv pe echipe de *knowledge workers* este cea a proiectelor software. Secțiunile următoare prezintă o analiză amănunțită a acestei categorii de proiecte, analiză în urma căreia se pot identifica o serie de nevoi pe care proiectele software în particular și cele orientate pe manipularea cunoștințelor în general, le au în ceea ce privește managementul.

Analiza succesului proiectelor software

Unul dintre primele lucruri pe care le aflăm atunci când citim o lucrare despre gestiunea proiectelor din domeniul IT (*Information Technology*) este acela că proiectele informatice se deosebesc foarte mult de toate celelalte

proiecte așa-zis tradiționale. Chiar și atunci când subiectul principal al lucrării este legat de managementul proiectelor în general, se fac frecvent referiri directe cu privire la ineficiența anumitor metode atunci când avem de-a face cu proiecte *software*. Din păcate sunt puține cazurile în se analizează sau explică în ce anume constau acele deosebiri și se propun mai degrabă soluții practice, mai mult sau mai puțin eficiente.

Unul dintre cele mai intens citate rapoarte statistice cu privire la succesul proiectelor informatice este *Standish Group Chaos Report* (disponibil la *www.standishgroup.com),* care an de an publică rezultate statistice ce evidențiază ponderea proiectelor finalizate cu succes, a celor eșuate precum și a celor care au întâmpinat dificultăți majore până la finalizare. Criteriile care stau la baza realizării acestui raport sunt se referă la gradul de satisfacere a componentelor triplei constrângeri a proiectelor - scop, timp și bani - sau mai precis la:

- respectarea și implementarea cerințelor specificate;
- încadrarea în bugetul estimat inițial;
- respectarea termenelor de livrare agreate.

Proiectele de succes sunt considerate acelea care respectă toate aceste criterii, cele eșuate sunt cele care s-au stopat fără a mai fi finalizate din cauza nerespectării unuia sau mai multor criterii, iar cele finalizate reprezentă proiectele care au fost până la urmă terminate, rezultatul proiectelor fiind implementat cu condiția reevaluării și ajustării în mod semnificativ a unora dintre criterii.

Tabelul 1.1 prezintă câteva dintre rezultatele raportului de-a lungul timpului, începând cu rezultatul "șocant" din 1994, ce releva o rată de succes extrem de scăzută, și până în 2015. Și această statistică vine să confirme existența unei diferențe evidente între proiectele informatice și celelalte proiecte, un procent de eșec de 20-30% fiind de neconceput în oricare dintre celelalte domenii, de la cel al construcțiilor de clădiri sau mașini până la servicii medicale.

Desigur că se poate discuta mult pe marginea acestor statistici și a relevanței lor. Pentru că "proiect informatic" este un termen care acoperă un spectru destul de larg și eterogen de proiecte, e neclar ce fel de proiecte au fost luate în considerare. Între anii 1994 și 2000 de exemplu, s-au studiat în jur de 30000 de proiecte informatice doar din Statele Unite ale Americii. Pentru 2004 raportul precizează că au fost luate în considerare peste 50000

de proiecte din întreaga lume (58% SUA, 27% Europa, 15% rest), cei de la *Standish Group International* asigurându-ne că s-a respectat un echilibru cantitativ între proiectele mici, medii și mari.

An	Succes	Finalizat	Eșec
1994	16%	53%	31%
1996	27%	33%	40%
1998	26%	46%	28%
2000	28%	49%	23%
2002	34%	51%	15%
2004	29%	53%	18%
2007	35%	46%	19%
2009	32%	44%	24%
2010	37%	42%	21%
2011	29%	49%	22%
2012	27%	56%	17%
2013	31%	50%	19%
2014	28%	55%	17%
2015	29%	52%	19%

Tabelul 1.1. Rezultatele raportului CHAOS al *Standish Group International* în perioada 1994 – 2015 ()

În altă ordine de idei, este destul de restrictiv să măsurăm succesul unui proiect luând în considerare doar criteriile scop, timp și ban. Calitatea produsului sau a serviciului rezultat reprezintă la rândul său un criteriu important. De asemenea, există proiecte care nu au respectat cele trei criterii, dar care au fost considerate de către organizațiile în care s-au desfășurat ca fiind de mare succes pentru că au atras ulterior noi proiecte importante, după cum există și proiecte ce au excelat în toate cele trei direcții însă presiunea exercitată a făcut ca echipa de proiect să se destrame rapid după terminarea proiectului pierzându-se experiența și expertiza câștigate în domeniul proiectului respectiv.

Începând cu anul 2012 raportul vine cu informații agregate pe diferite tipuri de metodologii utilizate în gestionarea proiectelor. După cum se poate observa și din tabelul 1.2 ce conține date privind succesul proiectelor software analizate în 2015, metodologiile Agile ameliorează simțitor numărul de proiecte eșuate, însă ele nu pot fi aplicate cu succes pentru toate tipurile de proiecte informatice.

Metodă	Succes	Finalizat	Eșec
Agile	39%	52%	9%
Waterfall	11%	60%	29%

Tabelul 1.2. Analiza succesului proiectelor software per metodologii în 2015, conform *Standish Group International* [2]

Ce am dorit să scoatem în evidență până la acest punct este că există diferențe notabile între proiectele informatice și proiecte din alte domenii de activitate și că aceste diferențe par a influența negativ succesul acestora. Vom detalia în secțiunea următoare principalele aspecte ce diferențiază proiectele informatice de proiectele tradiționale, așa cum au fost prezentate în [3].

Caracteristicile proiectelor software

Softul este nemăsurabil

Din fericire sau din păcate (în funcție de perspectiva din care privim lucrurile), activitatea de dezvoltare a softului este una creativă. Nu există, așa cum se întâmplă în cazul altor activități, un nomenclator care să precizeze care este timpul uzual necesar pentru implementarea unei ferestre ce conține, să zicem, două liste derulante, un tabel (*grid*) și două butoane. În ciuda dezvoltării de instrumente sofisticate de generare automată a codului sau de implementarea diverselor biblioteci de clase sau *framework*-uri, fiecare proiect nou vine cu provocările proprii în ceea ce privește creativitatea.

Un exemplu edificator în acest sens este dat în [4] în care se face un experiment chestionând un număr de 44 de experți în legătură cu numărul de instrucțiuni care apar în codul de mai jos:

```
#define      LOWER  0
#define      UPPER  300
#define      STEP   20
main()
{
  int fahr;
  for (fahr=LOWER; fahr<=UPPER; fahr=fahr+STEP)
    printf("%4d %6.1 f\n", fahr, (5.0/9.0*(fahr-32)))
}
```

Figura 1.1. Program scris în limbajul C
de conversie din grade Fahrenheit în grade Celsius [4]

Răspunsurile acestora cuprind toate numerele între 1 și 9 inclusiv (11 experți au răspuns 6, alții 11 au răspuns 9, restul alegând ca răspuns un alt număr din interval)! Concluzia este imediată: dacă niște experți în domeniul *software* nu reușesc să cadă de acord asupra unei întrebări simple pe baza unui program de 9(!) linii de cod, înseamnă că este de așteptat ca estimările într-un proiect software cu mii de linii de cod să difere foarte mult de la o persoană la alta, aceste estimări având o relativ restrânsă legătură cu experiența acumulată.

O practică uzuală este aceea ca estimările date (atunci când este vorba de estimări de timp și nu în puncte de complexitate) să fie mărite cu 20% de către managerul de proiect înainte ca ele să fie transmise către client, pentru a se asigura un oarecare "confort" (deși sunt situații în care acest 20% este departe de a fi suficient).

O altă consecință directă a acestui aspect o reprezintă dificultatea gestionării schimbărilor într-o echipă de proiect, persoanele care vor înlocui anumiți membri ai echipei rareori respectând estimările date de aceștia.

Și pentru ca lucrurile să fie "complete", pentru multe dintre activitățile estimate este foarte dificil de controlat progresul. Cele mai reprezentative exemple aici sunt activitățile de analiză și proiectare unde putem știi când s-au terminat aceste activități, dar nu vom putea specifica dacă la un moment dat ne aflăm la 50% la 75% din activitatea respectivă.

Softul este invizibil și intangibil

Rezultatul muncii unei echipe ce dezvoltă un produs informatic este compus dintr-o colecție de "texte" de diferite tipuri: documente de analiză și proiectare, cod sursă, manuale de utilizare și operare etc. Doar o parte dintre acestea interesează sau au vreo semnificație pentru cei care vor utiliza produsul.

Clientul final tinde să evalueze un produs informatic după ceea ce vede, iar ceea ce vede este în general o interfață grafică cu utilizatorul.

Deoarece nu există nimic concret de arătat clientului în faza de analiză a cerințelor, acesta își formează o imagine proprie asupra rezultatului final care de cele mai multe ori nu se potrivește cu produsul dezvoltat. În plus, există tendința de a include în cadrul specificațiilor elemente care reprezintă mai degrabă dorințe decât nevoi reale de *business*. Se ajunge astfel într-un punct caracterizat de un grad ridicat de frustrare, în care echipa de proiect știe că a dezvoltat conform specificațiilor funcționale dar clientul nu poate utiliza aplicația finală pentru că nu este ceea ce și-a imaginat că va fi. Acest rezultat poate fi ameliorat prin prezentarea unuia sau mai multor prototipuri în fazele timpurii ale dezvoltării produsului, însă clientul nu va face întotdeauna diferența între acestea și produsul final crezând că nu s-a depus un efort semnificativ între timp, interfața cu utilizatorul rămânând aproape neschimbată.

Tot din cauza invizibilității softului și implicit a complexității acestuia, cerințele inițiale par foarte ușor de modificat și implementat în cadrul unei aplicații.

Softul are o complexitate ridicată

Fără doar și poate produsele *software* au o structură extrem de complexă. Într-o aplicație de dimensiuni medii există extrem de multe conexiuni între diverse elemente *software* care fac aproape imposibilă testarea și validarea completă a acestora. Un exemplu simplu în care folosim nouă condiții succesive intercalate poate rezulta într-un număr impresionant de căi posibile care trebuie testate fiecare în parte pentru o validare completă. Acest lucru însă ar putea să nu fie realizabil nici într-o săptămână, chiar dacă testul fiecărei căi ar dura o secundă. Mai mult, aceste teste ar trebui reluate de fiecare dată când se face o modificare. Prin urmare multe dintre

defecte persistă o vreme îndelungată (chiar ani de zile) într-o aplicație până să fie descoperite sau, mai rău, până să își arate efectele.

După cum se spunea în aceeași lucrare menționată anterior: "*...în cazul proiectelor software așa numitele proceduri de control al calității au uneori de-a face mai degrabă cu limitarea defecțiunilor decât cu garantarea calității produsului final.*" [4]. Și acest lucru a rămas valabil și în ziua de azi, în ciuda tuturor metodologiilor moderne de dezvoltare a softului apărute în ultimii ani.

Complexitatea softului este dată și de numărul de tranziții și "traduceri" ce trebuie realizate între fazele ciclului de viață al unui produs. Specificațiile funcționale (redactate în limbaj natural) sunt translatate într-un model de analiză (diagrame vizuale care modelează toate soluțiile posibile ale problemei enunțate în specificații), care ulterior este translatat într-un model de proiectare (unde se particularizează modelul de analiză pentru o soluție specifică), care mai apoi este translatat în cod sursă, acesta din urmă fiind translatat într-un model executabil (prin compilare sau interpretare). Tot acest lanț de translatări, în ciuda faptului că anumite tranziții sunt automatizate, face procesul de dezvoltare a softului mult mai vulnerabil la erori umane. Acest lucru este cauzat de greșeli de "traducere" sau prin persistarea erorilor nedescoperită la timp în fazele de specificare sau analiză până în modelul executabil.

Softul este dinamic

Dinamismul cu care se confruntă proiectele software se manifestă în trei direcții relevante. Pe de o parte există atracția noutății tehnologice. Știm că tehnologia avansează foarte rapid și an de an apar biblioteci și *framework*-uri noi, și versiuni îmbunătățite ale limbajelor și mediilor de dezvoltare. Din punct de vedere al unui manager de proiect, păstrarea *framewok*-urilor inițiale în care a fost dezvoltat un anumit soft este o condiție elementară de păstrare a stabilității acestui soft. Pe de altă parte nu trebuie ignorată apetența echipei pentru a lucra cu ultimele tehnologii (inerent mai instabile și cu neajunsuri încă nerezolvate sau nediscutate pe forumurile de specialitate). Există un efort constant și care nu trebuie neglijat, de păstrare a unui echilibru între stabilitate și eliminarea unui sentiment de plafonare a echipei de dezvoltare.

A doua direcție ce conferă dinamism proiectelor software este fluctuația personalului, care în domeniul IT este foarte ridicată. O fluctuație "sănătoasă" pentru o organizație se situează undeva în jurul a 10 procente. Un studiu realizat de Ziarul Financiar arată ca în domeniul IT în România fluctuația de personal este la nivelul de 20%, și ea nu se datorează în principal nivelului de salarizare (cum este în cazul altor domenii) ci diferențelor de cultură și a sentimentului de nerealizare profesională. Efectele acestei fluctuații, după cum subliniam și mai devreme, constau în reconsiderarea planificărilor anterioare și de adaptare a noilor veniți la proiect. Nu trebuie însă neglijată tendința noilor veniți de a respinge codul scris de predecesorii lor, încercând uneori chiar să își asume responsabilitatea înlocuirii complete a acestui cod, acțiune ce conduce la anularea validărilor anterioare.

În fine, o dinamică aparte față de alte proiecte o au cererile de modificare frecvente venite din partea clientului în diverse faze ale ciclului de viață a dezvoltării unui proiect software. Această caracteristică a proiectelor software a condus la dezvoltarea metodologiilor Agile care includ acest aspect ca pe o componentă activă a procesului de dezvoltare.

2. Filozofia Agile

Manifestul Agile

Particularitățile proiectelor software descrise în capitolul precedent au condus în timp la dezvoltarea și implementarea unor practici care să asigure obținerea unei rate mai mari de succes. Astfel începând cu anii 1990 s-au creat mai multe metode independente de gestionare a proiectelor software. Aceste metode nu foloseau în mod necesar tehnici mult diferite față de cele utilizate în managementul tradițional însă se bazau pe un set de valori și principii derivate din particularitățile proiectelor software. Pe măsură ce astfel de metode independente se înmulțeau a apărut ca evidentă necesitatea găsirii unui numitor comun sau, de ce nu, a definirii unor standarde.

În 2001 un număr de 17 consultați independenți din industria software s-au întâlnit la un resort de schi din Utah, Statele Unite ale Americii, unde au pus bazele a ceea ce urma să se numească ulterior mișcarea Agile. Ei au elaborat următorul manifest ce se poate găsi tradus în peste 65 de limbi la adresa http://agilemanifesto.org și care sună astfel:

„Noi descoperim modalități mai bune de dezvoltare de software din experiența proprie și ajutându-i pe cei din jurul nostru în această activitate. Astfel am ajuns să apreciem:

***Indivizii și interacțiunea** înaintea proceselor și instrumentelor*

***Software funcțional** înaintea documentației vaste*

***Colaborarea cu clientul** înaintea negocierii contractuale*

***Receptivitatea la schimbare** înaintea urmăririi unui plan*

Bibliografie

[1] P. F. Drucker, The Landmarks of Tomorrow: A Report on the New "post-modern" World, Transaction Publishers, 1996.

[2] Standish Group International, „The CHAOS Report 2015," Boston MA, 2015.

[3] D. M. Suciu, „Particularități ale proiectelor informatice," *Today Software Magazine,* nr. 8, pp. 11-13, 2012.

[4] M. Norris, M. Payne și P. Rigby, The Healthy Software Project: a guide to successful development, John Wiley & Sons, 1993.

[5] ***, „Manifesto for Agile Software Development," 2001. [Interactiv]. Available: http//agilemanifesto.org. [Accesat 2018].

[6] D. Snowden, „Complex Acts of Knowing: Paradox and Descriptive Self Awareness," *Journal of Knowledge Management,* vol. 2, nr. 6, pp. 100-111, 2002.

[7] VersionOne, „The 10th Annual State of Agile Report," 2016. [Interactiv]. Available: https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf. [Accesat 2018].

[8] B. Boehm și R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Pearson Education Incorporated, 2003.

[9] M. Popendieck și T. Popendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley Professional, 2003.

[10] D. J. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press , 2010.

[11] V. Duarte, NoEstimates: How To Measure Project Progress Without Estimating, OikosofySeries, 2016.

[12] T. Demarco, Waltzing with Bears - Managing Risk on Software Projects, Dorset House, 2003.

[13] A. Kelly, Continuous Digital - An agile alternative to projects for digital business, 2017.

[14] PMI, PMBOK® Guide – Sixth Edition, Project Management Institute, 2017.

[15] J. Humble, The Case for Continuous Delivery, ThoughtWorks, 2014.

[16] J. McKendrick, How Amazon handles a new software deployment every second, zdnet, 2015.

[17] K. Obermüller și J. Campbell, Mob Programming - the Good, the Bad and the Great, "under the hood" blog, 2016.

[18] R. C. Martin, My Lawn, www.cleancoder.com, 2014.

[19] D. M. Suciu, „Product Mindset," *Today Software Magazine,* nr. 13, pp. 15-17, 2013.

[20] D. M. Suciu, „Best Practices în Agile," *Today Software Magazine,* nr. 15, pp. 13-15, 2013.

[21] D. M. Suciu, „Agile Humanum Est," *TodaySoftware Magazine,* nr. 39, pp. 12-14, 2015.

[22] D. M. Suciu, „Agile si adaptarea la schimbare," *Today Software Magazine,* nr. 63, pp. 11-13, 2017.

[23] J. P. Womack și D. T. Jones, Lean Thinking: Banish Waste and Create Wealth in Your Corporation, 2nd edition, Productivity Press, 2003.

[24] D. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press, 2010.

[25] C. Cobb, Making Sense of Agile Project Management, Balancing Control and Agility, John Wiley and Sons, 2011.

[26] A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley Professional, 2004.

[27] A. Cockburn, Agile Software Development: The Cooperative Game, 2nd edition, Upper Saddle River, NJ: Addison-Wesley, 2007.

[28] M. Cohn, Succeeding with Agile Software Development Using Scrum, Addison Wesley, 2010.

[29] M. Cohn, User Stories Applied: For Agile Software Development, Upper Saddle River, NJ: Addison-Wesley, 2004.

[30] S. Palmer și J. M. Felsing, A Practical Guide to Feature-Driven Development, Prentice Hall, 2002.

[31] E. Derby și D. Larsen, Agile Retrospectives, Making Good Teams Great, Pragmatic Bookshelf Publishing, 2006.

[32] A. S. Koch, Agile Software Development, Evaluating the Methods for Your Organization, Artech House, 2005.

[33] C. F. Kurtz și D. Snowden, „The new dynamics of strategy: Sense-making in a complex and complicated world," *IBM Systems Journal*, vol. 42, nr. 3, pp. 462-483, 2003.

[34] J. Langr și T. Ottinge, Agile in a Flash, Speed-Learning Agile Software Development, Pragmatic Bookshelf Publishing, 2011.

[35] C. Larman, Agile and Iterative Development, A Manager's Guide, Addison-Wesley, 2003.