ASPECTE PRIVIND DEZVOLTAREA „AGILĂ" A SOFTWARE-ULUI

Gabriela Rodica Hrin
Rodica.Hrin@ici.ro

Mihaela Tomescu
mtomescu@ici.ro

Institutul Național de Cercetare – Dezvoltare în Informatică, ICI, București

Rezumat: Dezvoltarea „agilă" a software-ului reprezintă un cadru de lucru conceptual, destinat realizării de proiecte de inginerie software. Acest cadru de lucru ia în considerare modificările care apar de-a lungul ciclului de viață al proiectului. Articolul prezintă o serie de aspecte definitorii privind dezvoltarea „agilă" a software-ului precum: principiile care stau la baza metodelor „agile", oportunitatea utilizării metodelor „agile", metode „agile" utilizabile în procesul de dezvoltare software, modelarea „agilă", procesul unificat „agil", procesul unificat dedicat întreprinderii, dezvoltarea adaptivă de software.

Cuvinte cheie: Metodologie „agilă", metode „agile", modelare „agilă", proces unificat „agil".

1. Concepte

Două definiții importante, ale tipurilor de metodologii de dezvoltare a software-ului, sunt prezentate în continuare:

- **metodologia „agilă"** (***agile***) este un sistem de metode destinat minimizării costului datorat modificărilor, în special într-un context în care apar evenimente neprevăzute în faze târzii ale desfășurării unui proiect sau în care trebuie efectuată adaptarea la factori incontrolabili importanți;
- **metodologia „non-agilă"** (***non-agile***) este, prin comparație cu metoda „agilă", o metodologie care urmărește obținerea eficienței prin anticiparea, controlul sau înlăturarea variabilelor astfel încât să se elimine necesitatea efectuării de modificări și costurile asociate modificărilor.

Dezvoltarea „agilă" a software-ului reprezintă un cadru de lucru conceptual, destinat realizării de proiecte de inginerie software. Acest cadru de lucru ia în considerare modificările care apar de-a lungul ciclului de viață al proiectului.

Majoritatea metodelor „agile" încearcă să minimizeze riscurile prin dezvoltarea de software în perioade scurte de timp, denumite iterații care durează, de obicei, de la o săptămână până la patru săptămâni. Fiecare iterație poate fi considerată un proiect mic de dezvoltare software de sine-stătător și conține toate sarcinile necesare dezvoltării unei funcționalități noi: planificarea, analiza cerințelor, proiectarea, codificarea, testarea și elaborarea documentației. Deoarece este posibil ca o iterație să nu adauge suficientă funcționalitate pentru garantarea realizării produsului, un proiect „agil" de dezvoltare software are ca obiectiv oferirea unui software nou, la sfârșitul fiecărei iterații. În cele mai multe cazuri, la sfârșitul fiecărei iterații, este furnizat un software. Acest fapt este, în mod special, adevărat atunci când software-ul este bazat pe Web și poate fi produs și lansat cu ușurință. Indiferent de situație, la sfârșitul fiecărei iterații, echipa proiectului reevaluează prioritățile proiectului.

Metodele „agile" pun accentul mai ales pe comunicarea în timp real, de preferat față – în - față, și nu pe documentele scrise. Majoritatea echipelor „agile" se află într-o anumită locație, sunt pregătite să intre în acțiune oricând și sunt formate din persoanele necesare realizării și finalizării software-ului. Minimum de persoane necesar pentru acest scop, este constituit de programatori și clienții lor (clienții sunt persoanele care definesc produsul și care pot fi manageri de proiecte, analiști de afaceri sau clienți efectivi). În această locație, se pot afla și alți membri ai echipei: testatori, proiectanți ai interacțiunilor, elaboratori ai documentației tehnice și manageri.

Metodele „agile" consideră versiunea de software (*working software*) ca fiind măsura principală a progresului obținut. Acest fapt, împreună cu preferința pentru comunicarea față – în – față, au ca rezultat faptul că metodele „agile" produc foarte puțină documentație scrisă în raport cu alte metode. În consecință, metodele „agile" sunt considerate ca fiind nedisciplinate.

2. Principiile care stau la baza metodelor „agile"

Metodele „agile" reprezintă o familie de procese de dezvoltare, nefiind doar o abordare singulară a dezvoltării de software. În anul 2001, 17 personalități din domeniul dezvoltării metodelor „agile" (denumite în acea perioadă „metode de dificultate redusă (lightweight)") au discutat la Snowbird, în Utah, despre modalitățile de creare a software-urilor într-un mod mai ușor, mai rapid și mai focalizat pe oameni.

Aceste personalități au creat „Manifestul metodelor „agile"", considerat pe scară largă ca fiind definiția canonică a dezvoltării „agile", precum și principiile „agile" pe care se bazează manifestul.

Principiile pe care se bazează manifestul metodelor „agile" sunt următoarele:

- prima prioritate este obținerea satisfacției clienților prin oferirea timpurie și continuă de software util;
- cerințele privind efectuarea de modificări sunt luate în considerare, chiar și în etapele târzii ale dezvoltării proiectului; procesele „agile" utilizează modificările în scopul creșterii competitivității clienților;
- sunt oferite versiuni de software cu o frecvență începând de la două săptămâni până la două luni, fiind preferate perioadele mai mici de timp;
- oamenii de afaceri și dezvoltatorii de software trebuie să lucreze împreună zilnic, de-a lungul întregului ciclu de viață al proiectului;
- proiectele se creează în jurul unor persoane motivate, cărora trebuie să li se asigure mediul și sprijinul de care au nevoie și să li se acorde încrederea că-și vor realiza corect sarcinile;
- cea mai eficientă și mai eficace metodă de transmitere a informațiilor în cadrul echipei de dezvoltare este comunicarea față – în – față;
- versiunea de software este măsura principală a progresului obținut;
- procesele „agile" promovează dezvoltarea durabilă;
- sponsorii, dezvoltatorii și utilizatorii trebuie să fie capabili să păstreze un ritm constant de lucru, pentru o perioadă nedefinită de timp;
- acordarea unei atenții continue excelenței tehnice și proiectării corecte mărește „agilitatea";
- simplitatea activităților de dezvoltare este esențială;
- cele mai bune arhitecturi, cerințe și proiectări sunt efectuate de echipe care se auto-organizează;
- periodic, echipa discută privind modul de eficientizare a activităților sale și își modifică comportamentul în conformitate cu concluziile rezultate.

Publicarea manifestului a declanșat o mișcare în industria de software, cunoscută sub denumirea de dezvoltare „agilă" a software-ului.

În anul 2005, Alistair Cockburn și Jim Highsmith au constituit un grup de lucru, format din experți în management, și împreună au elaborat un document cunoscut sub denumirea de „Declarația de independență a managementului proiectelor".

3. Oportunitatea utilizării metodelor „agile"

Deși metodele „agile" diferă în aplicațiile lor practice, au un număr mare de caracteristici comune, inclusiv dezvoltarea iterativă, și pun accentul pe interacțiune, comunicare și reducerea artefactelor intermediare cu resurse suplimentare.

Oportunitatea utilizării metodelor „agile" poate fi examinată din mai multe perspective. Din perspectiva produselor, utilizarea metodelor „agile" este adecvată atunci când cerințele apar brusc și se modifică rapid; utilizarea acestora este mai puțin adecvată pentru sistemele care au cerințe extrem de critice privind fiabilitatea și siguranța. Din perspectiva organizațională, utilizarea metodelor „agile" poate fi evaluată prin examinarea a trei dimensiuni cheie ale unei organizații: modul de lucru, persoanele implicate și comunicarea.

Pornind de la aceste dimensiuni, trebuie identificați factorii cheie de succes[1] și anume:

- modul de lucru al organizației trebuie să permită negocierea;
- trebuie să se acorde încredere persoanelor implicate;
- persoanele implicate trebuie să fie în număr relativ mic și mai competente;
- organizațiile se bazează pe informațiile pe care le furnizează dezvoltatorii;
- organizațiile au nevoie de un mediu care să faciliteze comunicarea rapidă între membrii echipei.

Cel mai important factor este dimensiunea proiectului. Cu cât dimensiunea proiectului este mai mare,

[1] Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. In: Advances in Computers, New York: Elsevier Science, pp. 1-66.

cu atât comunicarea față - în - față devine mai dificilă. Ca urmare, majoritatea metodelor „agile" sunt mai adecvate pentru proiecte cu echipe mici constituite din 20 – 40 persoane.

Dezvoltarea de software „agil" pe scară largă este, în continuare, un domeniu activ de cercetare.

O altă problemă serioasă care trebuie luată în considerare este aceea că ipotezele formulate inițial sau colectarea excesiv de rapidă a cerințelor pot duce la o deviere puternică de la soluția optimă, în special dacă clientul care definește produsul țintă nu știe prea multe despre necesitățile acestuia.

În mod similar, este posibil ca managerul echipei de dezvoltare să influențeze sau chiar să devieze proiectarea țintei într-o direcție care să nu fie adecvată proiectului.

Istoria a demonstrat că dezvoltatorii de software pot impune și, deseori, impun soluții unui client, în loc să-l convingă că soluția este adecvată necesităților acestuia, pentru ca, în final, clientul să descopere că nu poate utiliza soluția rezultată.

În teorie, natura rapid iterativă a metodei ar putea să limiteze acest fapt, trebuind însă să existe o reacție pozitivă sau negativă din partea clientului. Dacă nu există nicio reacție, eroarea ar putea să se extindă rapid.

Acest fapt poate fi evitat prin evidențierea cerințelor obținute într-o anumită fază și prin păstrarea legăturii cu clientul pe durata dezvoltării, implicându-l în permanență în testarea fiecărei noi versiuni realizate. În realitate, majoritatea clienților nu doresc să consume prea mult timp cu această activitate. De asemenea, acest fapt face dificilă asigurarea calității unui produs deoarece nu există obiective clare de testare care să nu se modifice de la o versiune la alta.

Pentru determinarea oportunității utilizării unei anumite metode „agile", este necesar să se efectueze o analiză mai sofisticată. De exemplu, metoda DSDM[2] oferă în acest scop un „filtru al oportunităților", în timp ce familia metodelor Crystal oferă criterii privind modul de selecție a metodei pentru un anumit proiect. Selecția se bazează pe dimensiunea, caracterul critic și prioritatea proiectului. Există metode „agile" care nu oferă, însă, astfel de instrumente destinate evaluării oportunității utilizării lor pentru un anumit proiect.

Despre unele metode „agile", precum DSDM și FDD[3], se afirmă că sunt adecvate pentru orice proiect de dezvoltare „agilă" de software, indiferent de caracteristicile contextuale[4].

O comparație între metodele „agile" arată că acestea susțin diferite etape ale ciclului de viață al dezvoltării software-ului în moduri diferite. Această caracteristică individuală a metodelor „agile" poate fi utilizată drept criteriu de selecție pentru alegerea metodelor „agile" candidate.

Dezvoltarea „agilă" a fost prezentată pe scară largă în multe documente (de exemplu, rapoarte de experimentare) ca fiind adecvată pentru echipe mici de dezvoltatori (mai puțin de 10 dezvoltatori) amplasate în același loc. Se consideră că dezvoltarea „agilă" este adecvată, în mod special, pentru echipele care se confruntă cu modificarea imprevizibilă sau rapidă a cerințelor.

Nu se poate garanta o eficiență ridicată în aplicarea dezvoltării „agile" în următoarele cazuri:

- implicarea unei echipe care conține mai mult de 20 de dezvoltatori;
- realizarea unei dezvoltări distribuite (echipe care nu sunt amplasate în același loc);
- există elemente critice privind misiunea și activitatea organizației;
- modul de lucru al organizațiilor este de tip „comandă – și – control".

Organizații, precum British Telecommunications (BT), au prezentat în diferite documente succesele unor proiecte realizate de dezvoltatori din Marea Britanie, Irlanda și India, care au utilizat metodologii „agile". Deși mai apar întrebări privind oportunitatea utilizării de metode „agile" pentru anumite tipuri de proiecte, se pare că dimensiunea etapelor de dezvoltare sau zona geografică nu reprezintă bariere în calea succesului.

Barry Boehm and Richard Turner propun utilizarea analizei riscurilor în scopul alegerii între metodele adaptive („agile") și cele predictive (conduse planificat). De asemenea, menționează că fiecare parte a întregului are domeniul său propriu de acțiune.

Domeniul metodelor „agile" se caracterizează prin:

---

[2] DSDM – Dynamic Systems Development Method – Metoda de dezvoltare dinamică a sistemelor.

[3] FDD – Feature Driven Development – Dezvoltare orientată spre caracteristici.

[4] Abrahamsson, P., Warsta, J., Siponen, M.T., & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis. În: Proceedings of ICSE'03, pp. 244-254.

- caracter critic scăzut;
- dezvoltatori experimentați;
- cerințe care se modifică foarte des;
- număr mic de dezvoltatori;
- mod de lucru care generează haos.

Domeniul metodelor conduse planificat se caracterizează prin:

- caracter critic înalt;
- dezvoltatori neexperimentați;
- cerințe care se modifică rar;
- număr mare de dezvoltatori;
- mod de lucru, care solicită desfăşurarea activităților în mod ordonat.

4. Metode „agile" utilizabile în procesul de dezvoltare software

Noțiunea de adaptare a metodei este referită în literatura de specialitate prin mai multe nuanțe precum: „adaptarea metodei conform necesităților", „adaptarea fragmentelor de metodă" şi „ingineria metodei contextuale".

„Adaptarea metodei conform necesităților" este definită astfel: „Un proces sau o capacitate în care agenții umani, prin modificări corespunzătoare referitoare la contexte, intenții şi fragmente de metodă şi prin interacțiuni dinamice între acestea, determină o abordare de dezvoltare a sistemului pentru o anumită situație a proiectului[5]".

Aproape toate metodele „agile" sunt adecvate pentru adaptarea metodei conform necesităților. Chiar şi metoda DSDM este utilizată în acest scop şi a fost adaptată cu succes conform necesităților într-un context CMM[6]. Adecvarea la situația abordată poate fi considerată o însuşire care face distincție între metodele „agile" şi metodele tradiționale de dezvoltare software, acestea din urmă fiind mult mai rigide şi prescriptive. Implicația practică a acestui fapt este aceea că metodele „agile" permit echipelor proiectului să adapteze practicile de lucru, în conformitate cu necesitățile proiectelor individuale. Practicile sunt activități şi produse concrete, care fac parte dintr-un cadru de lucru al metodei.

În cazul metodei Extreme Programming (XP), adaptarea metodei la necesități este făcută explicit. Una din ideile fundamentale ale metodei XP este aceea că nu există nici un proces care să fie adecvat pentru fiecare proiect ca atare, ci mai degrabă ar trebui adaptate practicile conform necesităților proiectelor individuale. Nu există rapoarte privind proiecte în care să fi fost adoptate toate practicile metodei XP, însă există rapoarte de adoptare parțială a practicilor XP.

Se poate face distincție între „adaptarea statică a metodei" şi „adaptarea dinamică a metodei". Ipoteza cheie, care se află la baza „adaptării statice a metodei", este următoarea: contextul proiectului este dat la începutul proiectului şi rămâne nemodificat pe durata executării proiectului. În „adaptarea dinamică a metodei" se presupune că proiectele sunt situate într-un context emergent. Un context emergent implică faptul că un proiect trebuie să facă față factorilor neprevăzuți ce apar şi afectează condiții importante ale proiectului. În consecință, contextul proiectului nu rămâne nemodificat pe durata executării proiectului.

Metodele „agile" diferă, într-o mare măsură, din punct de vedere al modului în care abordează managementul de proiect. Pentru unele metode, există linii directoare privind managementul de proiect.

Sistemul PRINCE2 este recomandat ca fiind un sistem adecvat pentru managementul de proiect.

Unele dintre cele mai cunoscute metode „agile" de dezvoltare software sunt următoarele:

- Extreme Programming (XP);
- Scrum;
- Agile Modeling (Modelare „agilă");
- Adaptive Software Development (ASD[7]);

---

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

[6] CMM – Capability Maturity Model – Model al maturității capacităților

[7] ASD – Adaptive Software Development – Dezvoltare adaptivă de software.

- Crystal Clear şi alte metodologii Crystal;
- Dynamic Systems Development Method (DSDM - Metodă de dezvoltare dinamică a sistemelor);
- Feature Driven Development (FDD - Dezvoltare orientată spre caracteristici);
- Lean software development (Dezvoltare „lină" de software);
- Agile Unified Process (AUP[8] - Proces unificat „agil").

Alte abordări care pot fi considerate sunt următoarele:

- Agile Documentation (Documentație „agilă");
- ICONIX Process (Proces ICONIX);
- Microsoft Solutions Framework (MSF[9] - Cadru de lucru pentru soluții Microsoft);
- Agile Data Method (Metoda „agilă" orientate spre date);
- Database refactoring (Refactorizarea bazei de date).

5. Modelarea „agilă"

Modelarea „agilă" este o metodologie bazată pe practica destinată modelării şi documentării sistemelor informatice. Aceasta este constituită dintr-o colecție de valori, principii şi practici pentru modelarea software-ului, care pot fi aplicate într-un proiect de dezvoltare software, într-o manieră mult mai flexibilă decât cea oferită de metodele tradiționale de modelare.

Principiile şi valorile practicilor modelării „agile" sunt destinate micşorării defectelor constate în dezvoltarea „agilă" de software. Principiul de „maximizare a valorii pentru grupurile de interese" determină dezvoltatorul de software să colaboreze cu clientul în furnizarea documentației adecvate.

Principiul de „modelare împreună cu alții" intenționează să implice grupurile de interese ale proiectului, în acelaşi mod în care cum sunt implicați clienții proiectului, în procesul de modelare a procesului în scopul alinierii modelului cât mai mult la cerințele utilizatorului final.

Există o dependență importantă între comunicarea față - în - față şi colaborarea clienților. Modelarea „agilă" este dificil de implementat în situațiile în care sunt implicate echipe mari (mai mult de 10 persoane), membrii echipei nu se află în aceeaşi locație sau nu au aptitudinile necesare realizării proiectului.

6. Procesul unificat „agil" (AUP - Agile Unified Process)

Procesul unificat „agil" (AUP) este o versiune simplificată a procesului unificat rațional / logic (RUP[10]). AUP descrie o abordare simplă şi uşor de înțeles, dedicată dezvoltării de software de aplicații de afaceri, utilizând tehnici şi concepte „agile" specifice RUP. Dintre tehnicile „agile", utilizate de AUP pentru îmbunătățirea productivității, se pot menționa: proiectarea bazată pe testare (TDD[11]), modelarea „agilă" (Agile Modeling), managementul „agil" al modificărilor şi refactorizarea bazei de date.

6.1. Fazele specifice AUP

AUP, la fel ca RUP, are patru faze:

- **lansarea:** constă în identificarea obiectivului inițial al proiectului şi a unei arhitecturi posibile a sistemului, precum şi în obținerea finanțării inițiale a proiectului şi aprobării de către grupurile de interese;
- **elaborarea:** prezintă arhitectura sistemului;
- **crearea:** reprezintă dezvoltarea de versiuni de software la intervale de timp prestabilite şi într-un mod incremental, care satisface necesitățile cu prioritatea cea mai mare ale grupurilor de interese ale proiectului;
- **tranziția:** validează şi pune în funcțiune sistemul în mediul de producție.

---

[8] AUP – Agile Unified Process – Proces unificat „agil".

[9] MSF – Microsoft Solutions Framework – Cadru de lucru pentru soluții Microsoft.

[10] RUP – Rational Unified Process – Proces unificat rațional / logic.

[11] TDD – Test driven design – Proiectare bazată pe testare.

6.2. Procesele specifice AUP

RUP defineşte nouă procese ale proiectului:

- modelarea afacerilor;
- obținerea cerințelor;
- analiza şi proiectarea;
- implementarea;
- testarea;
- punerea în funcțiune;
- configurarea şi managementul modificărilor;
- managementul proiectului;
- asigurarea mediului necesar desfăşurării activităților proiectului.

AUP defineşte, spre deosebire de RUP, numai şapte procese ale proiectului:

- **modelarea afacerilor**: în acest proces, sunt înțelese afacerile organizației şi este identificată o soluție viabilă, care să abordeze domeniul problemelor;
- **implementarea**: modelul sau modelele rezultate din procesul anterior sunt transformate în cod executabil şi se efectuează o testare la nivelul de bază, în special, la nivel de organizație;
- **testarea**: este efectuată o evaluare obiectivă în scopul asigurării calității; în această evaluare sunt găsite defecte, se validează faptul că sistemul funcționează conform proiectării şi se verifică dacă sunt satisfăcute cerințele;
- **configurarea şi managementul modificărilor**: în acest proces, este administrat accesul la artefactele proiectului; se efectuează urmărirea în timp a versiunilor de artefacte, precum şi controlul şi administrarea modificărilor acestora;
- **managementul proiectului**; acest proces constă în conducerea activităților desfăşurate pe durata proiectului; sunt efectuate administrarea riscurilor, conducerea persoanelor (atribuirea de sarcini, urmărirea progresului obținut etc.) şi coordonarea între persoane şi sistemele exterioare obiectivului proiectului pentru a se asigura livrarea la timp şi în limitele bugetului proiectului;
- **asigurarea mediului necesar desfăşurării activităților proiectului**: este susținut efortul depus în cadrul proiectului pentru asigurarea punerii la dispoziția întregii echipe a proceselor, îndrumărilor (standarde şi linii directoare) şi instrumentelor (hardware, software etc.) adecvate.

6.3. Principiile pe care se bazează AUP

Procesul unificat „agil" (AUP) se bazează pe următoarele principii:

- cunoaşterea de către persoanele echipei a ceea ce lucrează;
- simplitate;
- „agilitate";
- focalizare pe activități cu valoare înaltă;
- independența instrumentelor;
- utilizarea AUP în scopul satisfacerii necesităților clienților.

7. Procesul unificat dedicat întreprinderii (EUP – Enterprise Unified Process)

Procesul unificat dedicat întreprinderii (EUP[12]) este o extensie a RUP.

7.1. Fazele specifice EUP

EUP defineşte, în plus față de cele patru faze definite de RUP, încă două faze:

---

[12] EUP – Enterprise Unified Process – Procesul unificat dedicat întreprinderii.

- producția;
- retragerea de pe piață.

7.2. Procesele specifice EUP

EUP definește suplimentar față de cele nouă procese definite de RUP următoarele procese:

- un proces specific proiectului:
  - operații și suport.
- șapte procese specifice întreprinderii:
  - modelarea afacerilor întreprinderii;
  - managementul portofoliului întreprinderii;
  - arhitectura întreprinderii;
  - reutilizarea strategică;
  - managementul resurselor umane;
  - administrarea întreprinderii;
  - îmbunătățirea proceselor software.

7.3. Practici specifice EUP

Cele mai bune practici oferite de EUP sunt următoarele:

- dezvoltarea în mod iterativ;
- administrarea cerințelor;
- crearea arhitecturii;
- modelarea;
- verificarea continuă a calității;
- administrarea modificărilor;
- dezvoltarea în mod colaborativ;
- luarea în considerare a problemelor din afara activităților de dezvoltare;
- oferirea de versiuni de software la intervale regulate de timp;
- administrarea riscurilor.

8. Dezvoltarea adaptivă de software

Dezvoltarea adaptivă de software (ASD[13]), efectuată de Jim Highsmith și Sam Bayer, este un proces de dezvoltare de software, care a rezultat în urma activității de realizare rapidă a aplicațiilor. ASD încorporează principiul conform căruia starea normală de desfășurare a dezvoltării de software constă din adaptarea continuă a proceselor de dezvoltare.

ASD înlocuiește ciclul în cascadă, cu serii repetitive de cicluri de efectuare de presupuneri, colaborare și învățare. Acest ciclu dinamic permite învățarea și adaptarea continuă la starea emergentă a proiectului. Un ciclu de viață ASD este focalizat pe misiunea proiectului, bazat pe caracteristici, iterativ, dezvoltat în perioade scurte de timp, administrat din punct de vedere al riscurilor și tolerant la modificări.

Efectuarea de presupuneri se referă la paradoxul planificării și anume: este mai probabil să se presupună că toate grupurile de interese greșesc privind anumite aspecte ale misiunii proiectului atunci când încearcă să o definească.

Colaborarea se referă la eforturile de echilibrare a activității bazată pe părți predictibile ale mediului (planificarea și ghidarea lor) și adaptarea la modificările nesigure, cauzate de diferiți factori: tehnologie, cerințe, grupuri de utilizatori, vânzători de software etc.

Ciclurile de învățare, dedicate tuturor grupurilor de interese, se bazează pe scurte iterații constituite din

[13] ASD – Adaptive Software Development – Dezvoltarea adaptivă de software.

etape de proiectare, construire şi testare. Pe durata acestor iteraţii, se obţin cunoştinţe prin efectuarea de mici greşeli bazate pe false presupuneri şi corectarea acestor greşeli ceea ce duce la creşterea experienţei şi la o mai bună rezolvare a problemelor.

Bibliografie

1. http://en.wikipedia.org/wiki/Agile_software_development.
2. **BECK, K.**: Extreme Programming Explained: Embrace Change, Addison-Wesley, Boston, MA, 1999. ISBN 0-321-27865-8.
3. * * * Agile Manifesto principles.
4. **BOEHM, B., R. TURNER**: Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley, Boston, MA, 2004, Appendix A, pp. 165-194. ISBN 0-321-18612-5.
5. **LAPLANTE, P.A., C.J. NEILL**: „ACM Queue **1** (10), 2004. Retrieved on 2006-05-13.
6. **SOMMERVILLE, IAN**: [1982] (2007). 4.1.1. The Waterfall Model. Software Engineering, 8th edition, Harlow: Addison Wesley, p. 66f.
7. **COHEN, D., M. LINDVALL, P. COSTA**: An Introduction to Agile Methods. Advances in Computers, Elsevier Science, New York, 2004, pp. 1-66.
8. * * * Agile Processes Workshop II Managing Multiple Concurrent Agile Projects. Washington: OOPSLA 2002.
9. * * * Super size Me, Dr. Dobb's Journal, February 15, 2006.
10. **ABRAHAMSSON, P., J. WARSTA, M. T. SIPONEN, J. RONKAINEN**: New Directions on Agile Methods: A Comparative Analysis. Proc. of ICSE'03, 2003, pp. 244-254.
11. **BOEHM, B.; R. TURNER**: Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley Boston, MA:, 2004. ISBN 0-321-18612-5.
12. **SCHAAF, R.J.**: Agility XL. Systems and Software Technology Conference 2007, Tampa, FL.
13. * * * Bridging the Distance.
14. * * * Using an Agile Software Process with Offshore Development.
15. **AYDIN, M.N., F. HARMSEN, K. SLOOTEN, R.A. STAGWEE.**: An Agile Information Systems Development Method in use. Turk J Elec Engin, 2004, 12(2), pp. 127-138.
16. **ABRAHAMSSON, P., O. SALO, J. RONKAINEN, J. WARSTA**: Agile Software Development Methods: Review and Analysis. VTT Publications 478, 2002.
17. **AYDIN, M.N., F. HARMSEN, VAN K. SLOOTEN, R.A. STEGWEE**: On the Adaptation of An Agile Information Systems Development Method. Journal of Database Management Special issue on Agile Analysis, Design, and Implementation, 2005, 16(4), pp. 20-24.
18. * * * Agile Alliance at http://agilealliance.org/system/article/file/904/file.pdf.
19. **KURIAN, TISNI**: Agility Metrics: A Quantitative Fuzzy Based Approach for Measuring Agility of a Software Process. ISAM-Proc. of Int. Conf. on Agile Manufacturing '06 (ICAM-2006), Norfolk, U.S.
20. * * * sdmagazine.
21. * * * Extreme Programming Refactored.
22. * * * The Great Pyramid of Agile.
23. **WIEGERS, KARL E.**: Software Requirements 2: Practical Techniques for Gathering and Managing Requirements throughout the Product Development Cycle, 2nd ed., Redmond: Microsoft Press., 2003. ISBN 0-7356-1879-8.
24. **STELLMAN, A., J. GREENE**: Applied Software Project Management, O'Reilly Media, Cambridge, MA., 2005. ISBN 0-596-00948-8.