MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Gestiunea proiectelor software

Cuprins

1. Managementul proiectelor software

Din cele mai vechi timpuri, omul a simtit nevoia sa isi organizeze cat mai bine sarcinile si actiunile. Odata cu cresterea numarului de firme de software, a crescut implicit si numarul de proiecte. Fie ca este vorba de proiecte out-sourcing sau create intern, in cadrul acestor firme a existat nevoia unei mai bune planificari a proiectelor, a unei analize in timpul si dupa realizarea acestor proiecte. Pentru ca aceasta planificare sa fie cat mai buna, dar si pentru maximizarea rezultatelor, s-au dezvoltat cateva metode de dezvoltare a produselor software. Printre acestea, cele mai cunoscute sunt Agile/ SCRUM, Kanban, Waterfall (cascada) si metoda in V.

Unele dintre aceste metode urmaresc utilizarea cat mai buna a resurselor, altele cresterea productivitatii in timp a echipei cu scopul cresterii valorii acesteia, si implicit a randamentului. Indiferent de valorile si tehnica folosita, toate acestea sunt folosite pentru a putea maximiza productia si implicit veniturile.

Tehnicile mai moderne urmaresc atat cresterea echipei cat si a calitatii codului scris. Acest lucru este realizat prin implicarea echpei de testare in procesul de dezvoltare a produselor software. Din acest motiv pe langa echipa de dezvoltatori au aparut si alte echipe sau persoane conexe care sunt implicate in proiect. Printre acestea se numara si managerul de proiect, liderul de echipa, echipa de documentatie, echipa de prezentare a produselor, etc.

Desi nu exista o metoda de gestionare a proiectelor care sa se dovedeasca a garanta succesul acestora, multe firme adapteaza metodele consacrate conform nevoilor acestora, schimband anumite metodologii pentru a se plia modului lor de operare. De asemenea de multe ori se folosesc elemente apartinand mai multor metode de gestionare a proiectelor.

1.2 Metoda waterfall (cascada)

Aceasta metoda este o metoda secventiala in cadrul careia progresul proiectului este vazut ca unul “curgator”, mai precis trecand pe rand prin fiecare din fazele ce o alcatuiesc. Acestea sunt faza de Concept, Analiza, Design (Proiectare), Constructie (realizare), Testare, Productie sau Implementare si Mentenanta.

Aceste faze trebuie urmate in aceasta ordine, iar nerespectarea ordinii duce la o crestere a costurilor proiectului sau chiar la nefinalizarea lui. Odata trecuta o etapa, aceasta este considerata finalizata si nu se mai poate reveni la ea. Singura abatere de la aceasta regula apare in cazul in care in faza de mentenanta se descopera erori in functionarea programului. In acest caz se revine la faza de productie si testare, urmand din nou implementarea la client.

Aceasta metoda isi are obarsia in industia constructiilor si a manufacturii. In aceste industrii, aparitia sau obligativitatea unei modificari post-productie aduce cu sine costuri ridicate, daca nu chiar imposibilitatea realizarii acestora.

Desi primele informatii referitoare la acest subiect apar in anul 1956, o descriere mai amanuntita apare doar in anul 1970 si este facuta de Winston Royce. Acesta a prezentat si o varianta imbunatatita a acestei metode care implunea existent feedback-ului chiar din fazele timpurii ale proiectului (de la design), prin aceasta asigurandu-se gasirea sau chiar eliminarea inaintea aparitiei a unor erori de codare sau de design.

Aceasta metoda, la fel ca multe altele a avut argumente pentru si impotriva ei. Printre argumentele in favoarea acesteia se poate aminti faptul ca timpul petrecut la inceputul proiectelor poate aduce o economie serioasa acestora din punct de vedere financiar. Spre exemplu daca design-ul unui anumit

proiect este prea dificil, sau imposibil de implementat, acest lucru poate fi rezolvat printr-o regandire a intregului design. Daca acest lucru se descopera la implementare, de multe ori este prea tarziu pentru a putea modifica ceva, costurile fiind prea mari. Astfel, filozofia centrala a metodei waterfall este aceea ca managerul proiectului trebuie sa se asigure ca o faza este 100% finalizata si functionala inainte de a trece la faza urmatoare. Specificatiile pentru proiect trebuie cunoscute inaintea realizarii design-ului. De asemenea design-ul trebuie realizat pana la cele mai mici detalii inainte ca implementarea sa inceapa.

Un alt punct forte al acestei metode este faptul ca pune accent atat pe scrierea codului sursa cat si pe documentele ce trebuie realizate (spre exemplu cele de specificatii). Acest lucru asigura faptul ca in cazul in care unul din membrii echipei pleaca, nu vor exista “gauri” de cunostinte, toata echipa stiind detalii legate de proiect.

Acest model implica de asemenea mai multa disciplina din partea membrilor, deoarece necesita realizarea unor documente, dar acest lucru aduce cu sine o reusita a proiectului.

Desigur, aceasta metoda a avut parte si de contestari, intrucat nu exista o metoda perfecta care sa multumeasca pe toata lumea. Unul dintre cele mai importante argumente impotriva acesteia a fost faptul ca specificatiile se pot schimba pe parcurs. Clientul poate sa aduca oricand noi specificatii intrucat el este beneficiarul final. Daca aceste modificari apar dupa faza de design, care se considera a fi finalizata, exista posibilitatea ca modelul initial al proiectului sa nu mai concida cu noile specificatii. Acest lucru duce la invalidarea primei etape de design si implicit la cresterea costului proiectului.

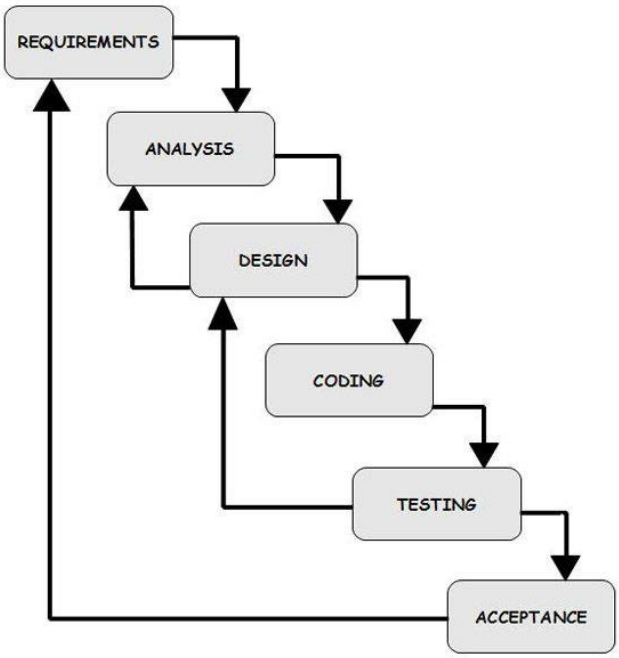

Fig. 1 – Metoda in cascada[6]

De asemenea exista posibilitatea ca designerii sa nu fie constienti de dificultatea implementarii anumitor componente din proiect. Este dificil sa astimeze de la inceput fiecare parte a proiectului, mai ales in cazul in care se folosesc tehnologii noi. Din aceasta cauza este de preferat schimbarea design-

ului, lucru care aduce cu sine o scadere a costului si o mai buna incadrare in bugetul alocat. Acest lucru insa contravine principiilor metodei in cascada (waterfall) deoarece se revine la o faza a proiectului dupa declararea acesteia ca fiind finalizata.

Extinzand putin acest proces, detinatorii de capital, care dealtfel sunt de cele mai multe ori persoane non-tehnice, nu cunosc tehnologiile folosite in dezvoltare. Din acest motiv exista posibilitatea ca design-ul sa implice folosirea minima a capacitatilor tehnice ale echipei.

Datorita unei intelegeri gresite a acestei metode, dar si a numarului ridicat de critici aduse acesteia de catre diferiti oameni abilitati din domeniu, ea a fost abandonata in special in proiectele guvernamentale. Acest lucru a dus la o scadere a popularitatii ei.

1.3 Metoda AGILE

Aceasta metoda se bazeaza pe dezvoltarea iterativa si incrementala a proiectelor software, in cadrul careia realizarea specificatiilor si implementarea solutiilor implica o colaborare stransa intre organizarea personala a fiecarui membru si existenta echipelor trans-functionale. Ea promoveaza ideea planificarii adaptive, dezvoltarea evolutiva (progrese vizibile de la un pas la altul), o abordare iterativa bazata pe cuante temporale si incurajeaza raspunsul la schimbare rapid si flexibil.

In februarie 2001, 17 programatori software s-au intalnit in Utah pentru a discuta legat de metodele de dezvoltare software. Cu aceasta ocazie au publicat « *Manifesto for Agile Software Development* «, care definea abordarea cunoscuta azi ca si Agile software development. Cativa dintre acestia au fondat o organizatie non-profit numita "Agile alliance" care promoveaza dezvoltarea proiectelor software respectand principiile din acest manifest. Principalele valori promovate de acesta sunt :

- Promovarea individului si a interactiunii dintre membrii in defavoarea respectarii stricte a unor procese
- Dezvoltarea de produse software functionale in defavoarea documentatiei exagerate
- Colaborarea cu clientul mai degraba decat negocierea contractului
- Reactionarea la schimbare in defavoarea respectarii stricte a unui plan

Exista cativa termeni respectiv cateva elemente particulare folosite in manifestul Agile. Acestea au fost definite inca de la inceputul metodologiei, atunci cand s-a redactat manifestul. De asemenea au fost alcatuite si 12 principii care stau la baza Agile, o parte din acestea fiind :

- Satisfactia clientului realizata prin livrarea rapida a partilor din proiect necesare acestuia
- Deschiderea spre schmbari in specificatii, chiar si intr-o faza avansata a dezvoltarii produsului
- Parti functionale din produsul final sunt livrate clientului regulat (la cateva saptamani)
- Dezvoltarea intr-un ritm sustinut si mentinerea unui ritm constant
- Cooperare intre echipa de teste, developeri si cei care stabilesc logica produsului
- Simplitatea- arta de a maximiza cantitatea de munca efectuata, este esentiala
- Echipe care se auto-organizeaza [4]

2. Metoda Scrum

Aparuta in anul 2001, aceasta metoda implica existenta a trei personaje necesare in cadrul fiecarei echipe care o adopta. **Scrum Master**, care se asigura ca procesul este respectat, ca impedimentele sunt inlaturate precum si ca fiecare membru al echipei are de lucru. **Product Owner**-ul (detinatorul produsului) este interfata intre client si echipa si **Echipa de dezvoltare** care se auto-organizeaza si care e responsabila de design si implementare.

Sprint-ul este unitatea de baza a dezvoltarii software in Scrum. Aceasta este o perioada care variaza intre o saptamana si o luna, depinzand de alegerea echipei si nu poate varia de la un sprint la altul. Fiecare sprint este precedat de o sedinta de planificare, in cadrul careia sarcinile pentru un sprint sunt definite si estimate ca si durata. La finalul sprint-ului exista o sedinta de retrospectiva cand se evalueaza mersul sprintului cu scopul perfectionarii echipei si a eliminarii eventualelor probleme aparute.

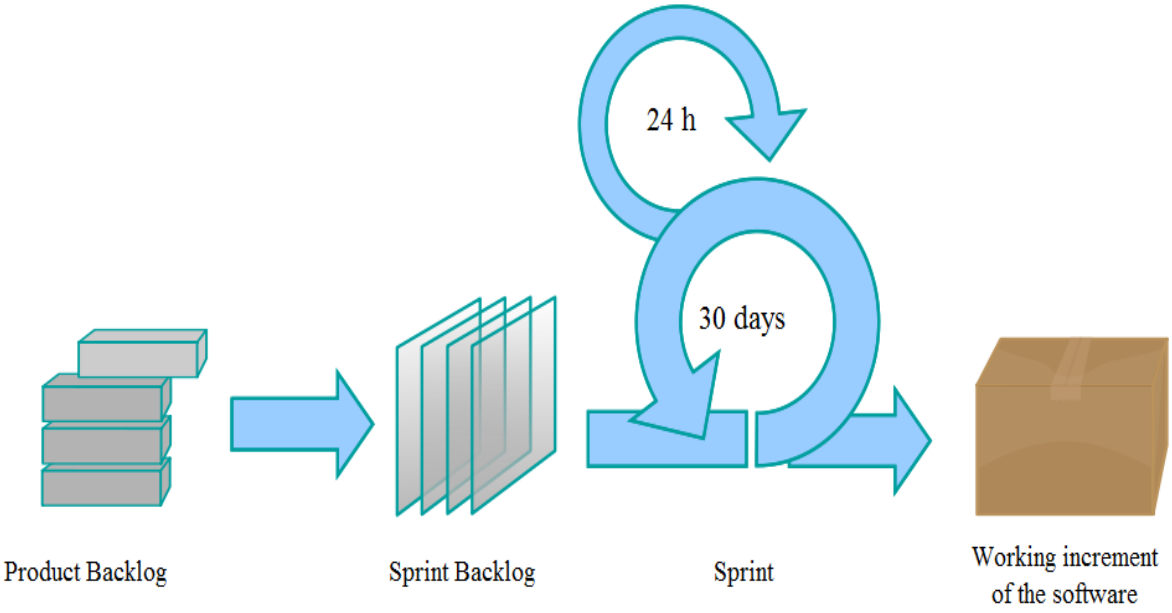

Fig. 2 – Elementele SCRUM[2]

La fiecare sprint, echipa creaza parti functionale din produsul final. Numarul de parti functionale ce sunt realizate in cadrul sprint-ului provine din *backlog*, mai precis o *lista prioritizata* a specificatiilor. Aceasta prioritizare este stabilita de product owner. Echipa stabileste cate dintre aceste specificatii pot sa intre intr-un sprint. Este important ca echipa sa faca acest lucru deoarece membrii acesteia sunt cei care stiu cat de repede pot sa termine o anumita sarcina. Odata inceput un sprint, lista de sarcini stabilita pentru aceasta perioada nu poate fi schimbata nici macar de catre detinatorul produsului. Motivul este usor de inteles, intrucat adaugarea de noi sarcini sau inlocuirea lor duce la modificarea numarului total de ore necesare realizarii lor si poate duce la imposibilitatea de a finaliza partea functionala stabilita spre a fi realizata in acea perioada. Daca din anumite motive exista sarcini care nu au fost finalizate in cadrul sprintului, acestea vor fi reintroduse in backlog si se vor executa in sprintul urmator. La sfarsitul acestei perioade, echipa va realiza o demonstratie a functionalitatilor implementate pentru client sau product owner. In acest fel se asigura faptul ca atat echipa cat si clientul stiu cu exactitate atat starea proiectului, cat si modul de functionare a produsului pana la acel punct.

Un principiu de baza al acestei metodologii este faptul ca un client isi poate schimba cerintele in ceea ce priveste functionalitatile necesare pentru produs. Din acest motiv, in Scrum se foloseste o abordare

empirica, acceptand faptul ca problema nu poate fi inteleasa sau definita pe deplin, focalizandu-se mai mult pe livrarea produselor si pe raspunsul prompt la specificatiile déjà existente.

2.1 Roluri

Exista trei roluri principale in procesul de dezolvate Scrum : Product Owner, Scrum Master si Echipa de dezvoltare.

Product owner-ul (detinatorul produsului) este vocea clientului si trebuie sa se asigure ca echipa livreaza un produs de valoarea dorita de client. El este cel responsabil de prioritizarea sarcinilor precum si de introducerea lor in backlog. Fara existenta acestui membru din echipa nu se poate aplica metoda scrum. De asemenea detinatorul produsului nu trebuie sa intre in dezvoltarea produsului, mai precis nu trebuie sa scrie cod. El este o persoana oarecum exterioara dezvoltarii.

Scrum master-ul este cel care faciliteaza intreg procesul. E responsabil de inlaturarea eventualelor impedimente. Acesta nu e lider de echipa, rol intalnit in procesele clasice de ierarhizare in cadrul companiilor, ci actioneaza ca o persoana tampon intre echipa si orice influente exterioare. In acest fel **echipa de dezvoltare** se va putea concentra strict pe aceasta parte, fara a fi deranjata de alte persoane. Acest rol a mai fost numit si lider-servitor in literatura de specialitate, tocmai cu scopul amplificarii rolului dual pe care acesta il are. Scrum master-ul nu are drept de veto, dreptul absolut avandu-l detinatorul produsului.

2.2 Sedinte

Exista *mai multe sedinte* care trebuie sa aiba loc pentru ca metodologia Scrum sa se desfasoare conform manifestului redactat de fondatorii acesteia. Aceste sedinte sunt : sedinta zilnica, de estimare, intalnirea scrum master-irlor, cea de planificare, retrospective si cea de analiza.

a)Sedinta zilnica nu trebuie sa dureze mai mult de 15 minute. Aceasta are loc in fiecare zi la aceasi ora, stabilita la inceputul proiectului impreuna cu echipa. Pot participa si persoane din alte echipe, dar de regula cuvantul este luat doar de catre membrii echipei intrucat acestia stiu cu exactitate starea proiectului. In timpul sedintei are loc « sincronizarea » echipei, mai precis fiecare membru al echipei trebuie sa raspunda punctual la intrebarile : ce a facut de la ultima sedinta, ce va face in continuare si ce probleme a intampinat. Pe langa faptul ca echipa afla starea exacta a proiectului, rolul acestei sedinte este si acela de a ghida oamenii care intampina anumite dificultati spre persoanele care ii pot ajuta. Sedinta este facilitata de scrum master.

b)Intalnirea scrum master-ilor are loc de regula dupa sedinta zilnica si are ca si scop sincronizarea cu celelalte echipe. Aceasta sincronizare ajuta la o buna integrare a diferitelor parti din produse intre ele. Intrebarile la care se raspunde sunt similare celor de la intalnirea zilnica, doar ca se reprezinta o echipa intreaga : ce a facut echipa de la ultima intalnire, ce va face in continuare si ce impedimente a intalnit (eventuale intarzieri cauzate de alte echipe).

c)Sedinta de planificare are ca si scop stabilirea prioritatilor si a sarcinilor ce vor intra in sprint-ul urmator. Acest lucru se realizeaza cu sprijinul si participarea tuturor membrilor echipei, inclusiv a detinatorului de produs. Are o limita superioara de 8 ore, insa de cele mai multe ori ea nu dureaza mai mult de 2-3 ore.

d)Sedinta de analiza si *de retrospectiva* are ca si scop realizarea unei demonstratii practice asupra functionalitatilor implementate in ultimul sprint. De asemenea se raspunde si la trei intrebari : ce a mers bine, ce nu a mers bine si ce se poate imbunatati. Aceste intrebari au ca si perioada

de analiza sprintul care s-a incheiat. Este important ca aceasta sedinta sa nu fie doar una formala. Doar analizand cu atentie cele intamplate echipa poate progresa si in felul acesta velocitatea echipei ajunge sa creasca de la un sprint la altul.

Pentru ca aceasta sedinta sa nu fie doar una formala, si ca echipa sa progreseze in timp este necesara alcatuirea unei e) *liste de actiuni* ce trebuie indeplinite. Aceste actiuni trebuie sa respecte principiul SMART. Mai precis ele trebuie sa fie Specifice, adica sa nu aiba un caracter general. Apoi ele trebuie sa fie masurabile. Acest lucru inseamna ca realizarea lor sa poata fi cuantificata intr-un fel. Spre exemplu un scop de tipul trebuie sa citesc mai mult nu este masurabil pe cand trebuie sa citesc trei pagini pe zi pentru a avea o cultura mai bogata este masurabil. De asemenea aceste actiuni trebuie sa fie realiste si sa poata fi duse la indeplinire si nu in ultimul rand trebuie sa fie orientate pe o durata de timp finita. Acest lucru este important pentru a putea sti la sfarsitul acelei perioade daca actiunea a fost indeplinita sau nu cu succes.

2.3 Artefacte

Exista o serie de artefacte care intra in componenta unui proiect SCRUM. Acestea sunt absolut necesare pentru buna realizare a oricarui proiect. Daca ele lipsesc, intreg procesul este de prisos, el avand nevoie de toate aceste artefacte.

2.3.1 Product backlog

Acesta se prezinta sub forma unei liste ordonate de specificatii pentru un anumit produs. Contine entitati denumite *PBI* (Product Backlog Item) care sunt ordonate in functie de prioritati de catre Product Owner, bazandu-se pe anumite criterii cum ar fi riscul, valoarea business, dependente, data la care o anumita componenta trebuie furnizata, etc. Elementele adaugate in product backlog sunt scrise de cele mai multe ori in format poveste. Product backlog reprezinta acel « ce« care trebuie realizat, sortat in ordinea relativa in care trebuie executat. El poate fi editat de catre orice membru al echipei, insa Product Owner-ul este cel care raspunde la final de ordonarea acestora, pentru echipa de dezvoltare. Acest artefact contine estimari grosiere cu privire la valoarea business si efortul necesar pentru dezvoltarea unui element. Aceste valori sunt exprimate in puncte valoare (story points) folosind o secventa Fibonacci. Estimarile ajuta product owner-ul sa realizeze linia cronologica pentru dezvoltarea produsului si poate influenta ordonarea listei. Spre exemplu daca functionalitatile de adaugat validari si adaugat support pentru tabele au aceasi valoarea business, cea cu valoarea efortului de dezvoltare mai mica va avea prioritate in cele mai multe cazuri deoarece rezultatul vizibil va surveni mai repede crescand totodata moralul echipei.

2.3.2 Sprint Backlog

Aceasta reprezinta o lista de sarcini pe care echipa de dezvoltare trebuie sa o realizeze pe parcursul sprint-ului urmator. Lista se alcatuieste selectand functionalitati din partea de sus a product backlog, pana la punctul in care echipa simte ca are suficient de lucru pentru a umple intreaga perioada a sprint-ului. Acest lucru se realizeaza punand echipei de dezvoltare intrebarea » putem sa facem de asemenea si acest lucru ?» iar daca raspunsul este favorabil se adauga functionalitatea in aceasta lista. Ca si puncte de referinta trebuie tinut cont de velocitatea echipei din sprint-urile anterioare. In cazul in care este primul sprint, cel mai probabil estimarea va fi dificila si marja de eroare va fi mai mare. Aceasta velocitate a echipei din sprint-urile trecute ajuta la stabilirea efortului ce poate fi depus in sprintul curent.

Aceste elemente din lista trebuie impartite in *task*-uri de catre echipa de dezvoltare. De regula aceste task-uri sunt cu o durata fizica de 4 pana la 16 ore, insa aceste limite pot fi depasite in cazul in care se considera necesar. Depasirea insa nu trebuie sa fie prea mare intrucat un numar mare de ore pentru un task inseamna de regula o granularitate prea mare pentru acesta. Cu un astfel de nivel de detaliu, echipa de dezvoltare ar trebui in mod normal sa stie exact ce trebuie sa faca, iar un task sa poata fi ales de catre orice membru al echipei. Task-urile nu se asigneaza niciodata la nivelul backlog-ului pentru sprint, ci acestea sunt luate de catre membri la sedintele zilnice, conform prioritatilor stabilite de product owner.

Acest artefact apartine membrilor echipei de dezvoltare si toate estimarile incluse sunt furnizate de catre acestia. Deseori se foloseste *o tabla* pentru a fi mai vizibile aceste task-uri, chiar daca exista unelte software special pentru acest lucru.

2.3.3 Graficul orelor arse

Graficul orelor arse in cadrul sprintului este un grafic public ce arata munca ce mai trebuie depusa in sprint-ul curent. El este actualizat in fiecare zi si ofera o vedere simpla a progresului echipei din punct de vedere al proiectului. Exista si alte tipuri de grafice care nu sunt specifice sprint-ului cum ar fi graficul functionalitatilor ramase pana la livrare, care arata cantitatea de munca ce mai trebuie realizata pentru a ajunge la un produs finalizat, graficul de erori realizate de echipa, etc.

3. Unelte software folosite pentru managementul proiectelor Agile

Pentru o mai buna organizare in cadrul echipei care adopta metodologia Agile/Scrum exista necesitatea folosirii anumitor unelte. Atunci cand bugetul nu permite investirea unei sume in produse software specializate pentru acest lucru se poate folosi **o simpla tabla**, cunoscuta si sub denumirea de **scrum board** (tabla de scrum) care este impartita in mai multe zone. *Prima parte* este cea cu *sarcinile din sprintul curent*, impartite la randul lor in *3 parti : de rezolvat, in progres si finalizate*. O *alta zona* include toate elementele din backlog prioritizate. Pe langa acestea mai pot exista *diverse alte zone*, cum ar fi cea a metricelor/graficelor sau a impedimentelor.

Desi intreg procesul Scrum se poate desfasura si in acest fel, este recomandata folosirea unor unelte software adecvate. In acest fel se usureaza mult munca scrum master-ului si in plus pot exista diferite unitati de masurare a velocitatii si performantei echipei.

Una dintre companiile fruntase in acest domeniu este JIRA. Produsul acestora, GreenHopper adauga ideea de Agile managementului proiectelor in stil clasic. Fie ca membrii echipei au cunostinte si experienta solida in domeniul Agile sau ca sunt incepatori, acest produs este perfect pentru vizualizarea proceselor curente si a stimularii imbunatatirii incrementale.[3]

Exista multe parti implicate in dezvoltarea unui proiect, de la echipa de design, de development, cea de asigurare a calitatii pana la cea de conducere si inclusiv clientul. Aceste unelte ajuta la o mai buna structurare si integrare a tuturor partilor implicate in proiect. Suita de produse Atlassian ofera numeroase unelte printre care Jira, Confluence, FishEye/Crucible, Clover, Bamboo etc.

3.1 Jira

Jira este o unealta software folosita pentru a monitoriza impedimentele aparute. Ea poate implica toate echipele amintite mai sus. Impreuna cu plugin-ul GreenHopper ofera o modalitate usoara de a monitoriza intreg procesul. Ca si structura produsul este gandit sa cuprinda *trei table/zone*. Zona de *planificare* cuprinde o privire de ansamblu asupra elementelor din backlog, planificarile pentru sprint-uri

si pentru finalizarea produsului precum si prioritizarea sarcinilor ce mai trebuie rezolvate. *Zona de task-uri* (sarcini) contine sarcinile particularizate pentru fiecare membru al echipei, statusul acestora in sprint-ul curent (in progres , de rezolvat sau finalizate) precum si anumite tichete sau mesaje particularizate in fiecare zi.

Zona metricilor si a graficelor este una din cele mai importante. Ea ofera informatii legate de starea proiectului in general, dar si particular pentru fiecare membru al echipei. Unul dintre cele mai folosite grafice este cel al sarcinilor finalizate din sprint-ul curent. Folosind acest grafic echipa isi poate da seama daca exista probleme privind velocitatea membrilor, dar si daca cele planificate la inceputul sprint-ului vor putea fi terminate la timp.[2]

Iata cateva dintre elementele cheie ale produsului, care pot sau nu sa fie folosite in proiect, in functie de necesitatile managerului si ale echipei:

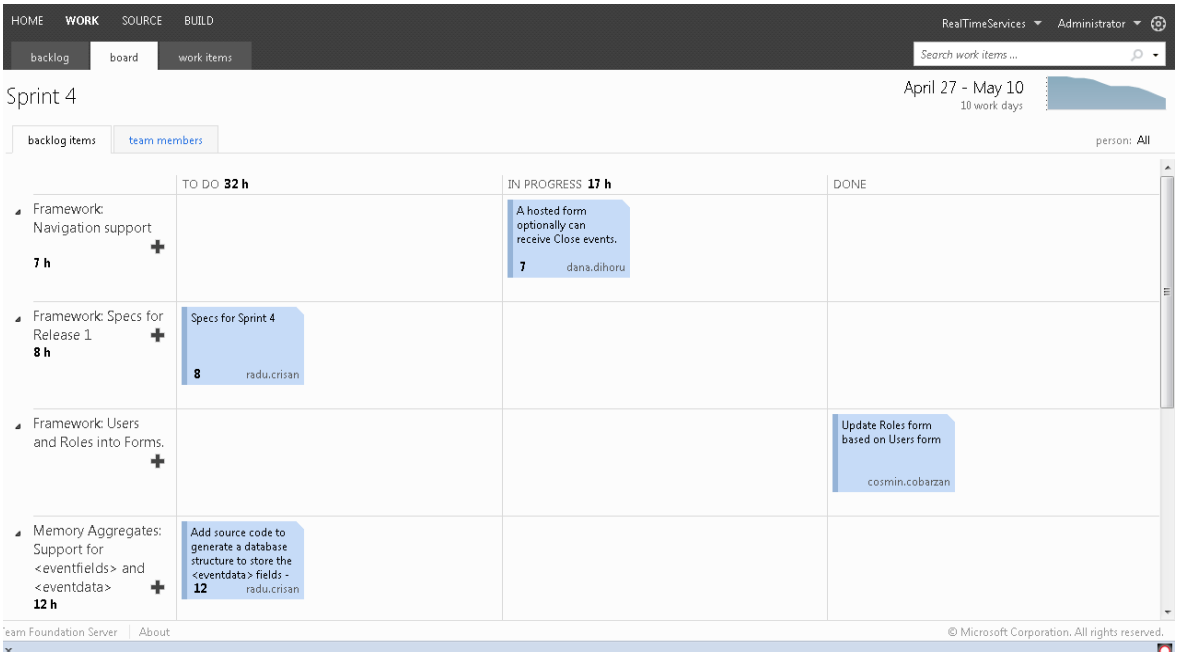

Fig. 3 – Scrum folosind Jira

- Managementul mai multor echipe - folosirea ierarhizarilor si a diferitelor alte metode ajuta la o buna intelegere a modului de functionare si implicit la o utilizare adecvata in asa fel incat sa poata fi urmarite mai multe echipe folosind aceasta unealta la ierarhizarea componentelor
- Ierarhizarea versiunilor
- Prioritizarea elementelor din backlog
- Semnalizarea vizuala a impedimentelor – prin introducerea unor elemente vizuale pe tichetele care necesita o abordare urgenta, acestea vor fi mai vizibile si implicit atentia se va putea concentra asupra lor

- Configurabilitatea elementelor si a workflow-ului de rezolvare a acestora
- Posibilitatea de a configura statutul tichetelor aparute
- Interpelari folosind AJAX – nu necesita reimprospatarea intregii pagini
- Diferite grafice, in functie de necesitati, chiar si customizabile – spre exemplu numarul de erori asociate unui membru al echipei
- Managementul datei de finalizare a produsului, aceasta nefiind fixa de la bun inceput[4]

Exista multe alte astfel de functionalitati, dar ele nu sunt folosite foarte des, ci doar atunci cand echipa simte necesara si utila folosirea acestora.

3.2 Scrum for Team Systems

Acest produs este disponibil impreuna cu sursele lui pe CodePlex. Este de fapt un add-in la Visual Studio Team Systems. A fost dezvoltat pe parcursul a patru ani de catre Crispin Parker. Pentru realizarea acestui proiect, autorul a renuntat la locul sau de munca pentru a se dedica in totalitate acestuia. Este intr-adevar un proiect complex pentru o persoana.

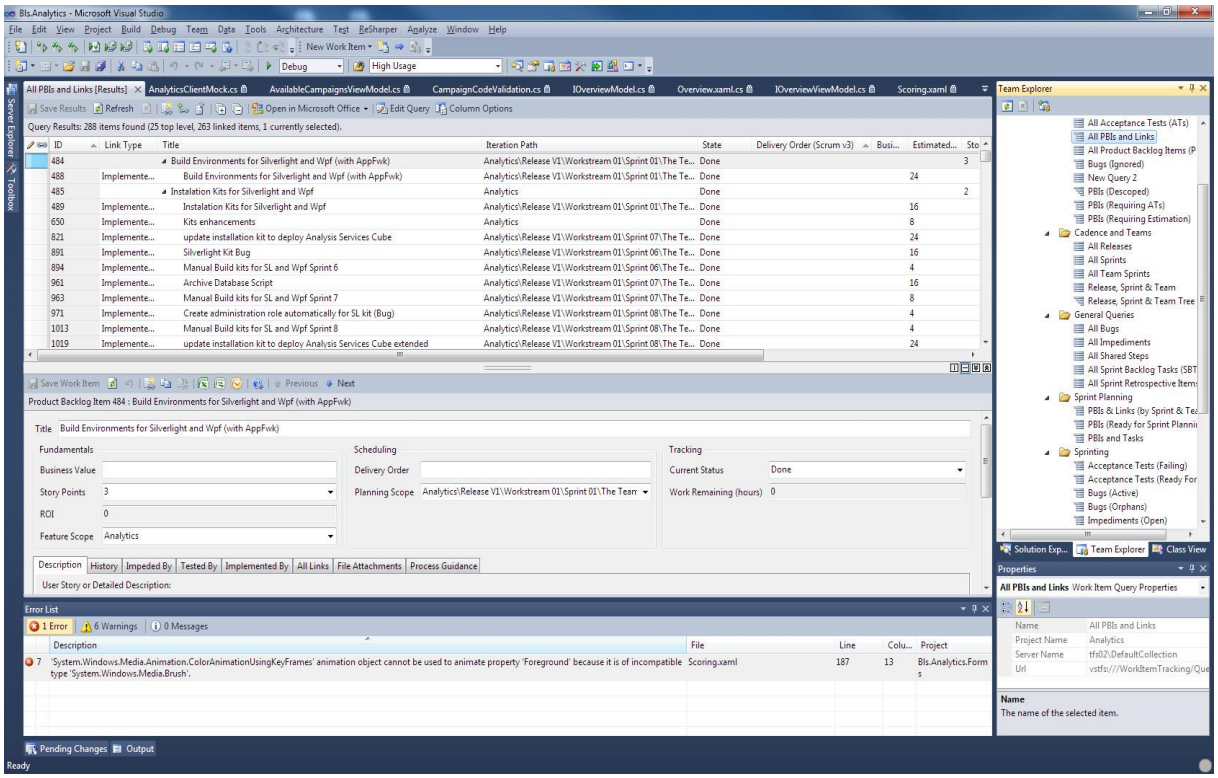

Fig. 4 – Team Explorer din Visual Studio 2010

Pentru acest proiect este necesar *Visual Studio 2010*, inclusiv *Team Explorer*. Modul de functionare al acestui program este destul de usor de inteles. Utilizatorii (membrii echipei) sunt de fapt utilizatorii de pe masina unde este instalat Team Explorer. In acest fel se asigura o buna integrare cu solutia dezvoltata. Fiecare sarcina are asociat un ID care se regaseste si pe Team Explorer. Desi din punct de vedere vizual nu prezinta foarte multe facilitati, totusi acest produs ofera functionalitatile de baza necesare. Astfel se pot vedea diferite rapoarte precum graficul de ardere al orelor efectuate, graficul

partilor functionale dezvoltate in functie de timp si alte metrici necesare pentru sedintele de retrospectiva sau pentru estimarea finalizarii proiectului.

Un dezavantaj din punctul de vedere al utilizatorului este faptul ca acesta este nevoit sa execute niste interogari mai complexe, chiar daca destul de usor de intuit, pentru a putea sa gaseasca anumite elemente asociate lui, de genul erorilor sau al sarcinilor pe care trebuie sa le efectueze.

Nici din punctul de vedere al tablei de scrum nu se prezinta intr-un mod prietenos pentru utilizatori acest produs. Din nou membrii echipei sunt nevoiti sa execute ei insisi interogari pentru a gasi sarcinile asociate lor si a le incadra in una din cele trei categorii: in progres, de rezolvat si finalizat. Tot la fel trebuie procedat si cu „scaderea numarului de ore ramase pentru o anumita sarcina.

Avantajul principal pentru acest produs este faptul ca este complet integrat cu Visual Studio. In acest fel toate elementele pot urma un flux normal, de la dezvoltare la testare si retestare, incluzand si partea de management al proiectelor. Chiar daca acesta se poate folosi gratuit, el este destul de util in special pentru firmele mici in care tehnologia SCRUM este destul de tanara din punct de vedere al utilizarii acesteia in cadrul proiectelor. Totusi dupa o anumita perioada si pentru a oferi mai mult profesionalism se recomanda folosirea altor produse mai performante.

4. Exemplu de proiect SCRUM

Pentru testarea Facilitatilor oferite de TeamScrum Pro s-a ales un proiect Analysis. Acest proiect a fost dezvoltat pe parcursul a trei luni de zile in sprinturi de cate doua saptamani. S-a ales acest interval din motive obiective, mai precis datorita faptului ca echipa avea deja o experienta in spate, iar membrii relativ noi ai echipei aveau o vechime de minim sase luni, astfel ca s-a putut stabili cu aproximatie velocitatea intregii echipe. Chiar daca nu se cunostea acest lucru, se putea determina aproximativ o medie. Din acest motiv s-au ales sprint-uri de doua saptamani, astfel incat parti functionale sa poata fi livrate in timp util.

Primul lucru necesar inaintea introducerii sarcinilor este *crearea de utilizatori*. Utilizatorii pentru TeamScrum sunt creati pe masina unde este instalat produsul. Astfel utilizatorii din LocalMachine devin si cei de pe TeamScrum. De asemenea produsul este integrat si cu Visual Studio 2011, astfel ca sarcinile se pot integra foarte usor cu TeamScrum. Practic pentru fiecare sarcina un tester poate sa realizeze TestCase-uri si sa o valideze sau nu. In acest fel se asigura faptul ca nu vor exista parti din aplicatie care sa nu fie cunoscute de testeri sau care sa nu fi fost testate din cauza lipsei de specificatii.

Dupa instalarea TeamScrum si integrarea acestuia cu VisualStudio se poate crea un nou proiect pe Team Explorer. Odata creat acest proiect, toate task-urile create in cadrul Team Explorer sunt automat legate de catre Visual Studio la acesta. In figura 4 se poate observa cum pentru produsul Analytics s-au creat zece elemente in cadru listeil PBI. Acestea au fost inpartite pe doua categorii: constructia infrastructurii programului si realizarea de functionalitati. In prima categorie au intrat elemente precum realizarea scriptului de build, realizarea kit-ului de instalare, realizarea scriptului de arhivare. In cea de-a doua categorie intra elemente precum realizare modul de autentificare, realizare forma de inregistrare utilizatori, creare serviciu WCF, Windows Communication Foundation, etc. Tot in aceasta forma se pot observa si detalii cu privire la unul dintre elementele din lista PBI. Astfel pentru task-ul de creare de script build se poate observa ca acesta contine 3 story points, statusul curent este in progres si ca are inca 6 ore pana la a fi finalizat.

Un lucru important pentru orice echipa atunci cand se realizeaza sedinta de planificare este cunoasterea capacitatii totale a echipei. Fiecare sprint va avea o capacitate diferita datorita implicarii membrilor in alte proiecte, a concediilor sau a lipsei acestora. Figura 5 prezinta felul in care se poate realiza aceasta alocare pentru un sprint.

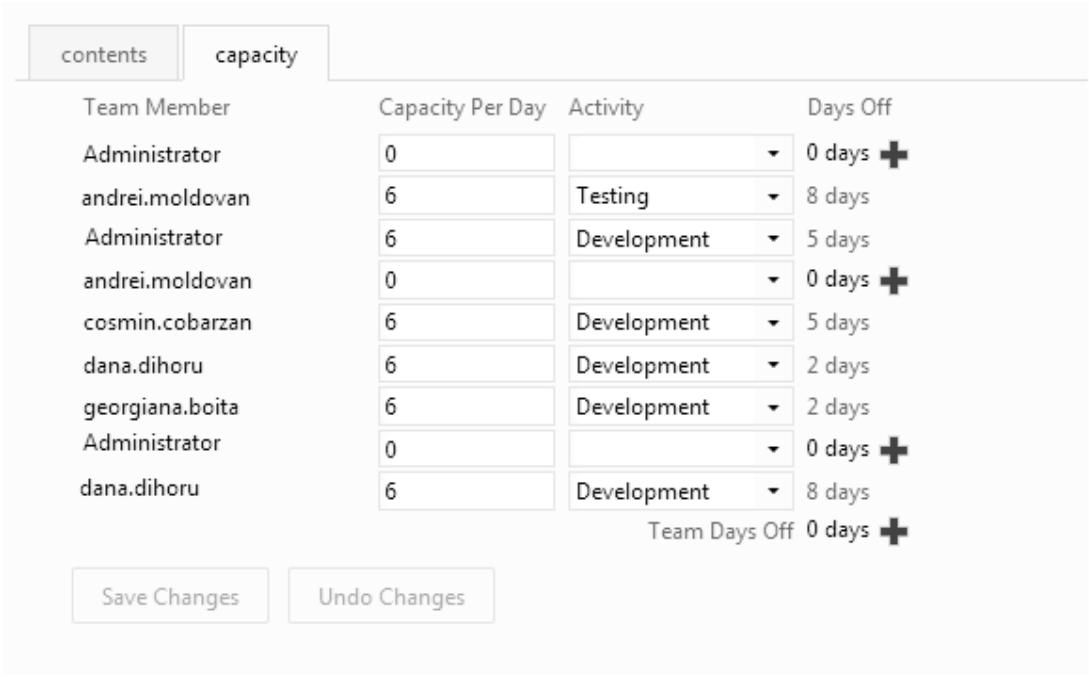

Fig. 5 – Alocarea membrilor echipei

In cazul de fata echipa este alcatuita din noua membri. Pentru fiecare membru se va seta numarul de ore zilnice pe care acesta il va petrece pentru proiect. Pentru a lasa o marja de eroare s-a ales un numar maxim de 6 ore (din opt posibile) in asa fel incat daca apr evenimente neprevazute ele sa nu afecteze calitatea sprintului. De asemenea se va putea seta activitatea: dezvoltare, testare, documentatie precum si numarul de zile libere pe care fiecare membru le are. Zilele de sambata si duminica vor fi considerate implicit libere. Numarul total de zile va fi calculat de catre Team System pe baza setarilor initiale de durata a sprintului.

4.2 Metrici

Unul dintre cele mai importante lucruri in SCRUM este reprezentata de metrici. Acestea indica mersul proiectului si poate indica din timp daca exista probleme in procesul de dezvoltare. Spre exemplu daca product owner-ul va observa o panta prea lina a graficului de ardere al orelor pentru sprintul curent va putea lua masuri in acest sens.

Pe langa *graficul de ardere al orelor* (sprint *burndown chart*), Team system ofera inca doua grafice importante: *user acceptance* tests si *bug history*.

Pentru burndown chart exista doua modalitati de accesare. Prima este chiar in pagina de inceput a proiectului in coltul din dreapta. S-a ales aceasta metoda intrucat acest grafic trebuie sa poata fi accesat usor si sa fie intr-un loc vizibil pentru a putea preintampina eventualele impedimente. Celalalt loc de unde poate fi accesat este din team explorer. Pentru randarea lui se foloseste Microsoft SQL Server Reporting Services. In cazul celui din urma se poate selecta si sprintul pentru care sa se genereze acesta, intrucat este realizat ca si un raport cu extensia .rdl.

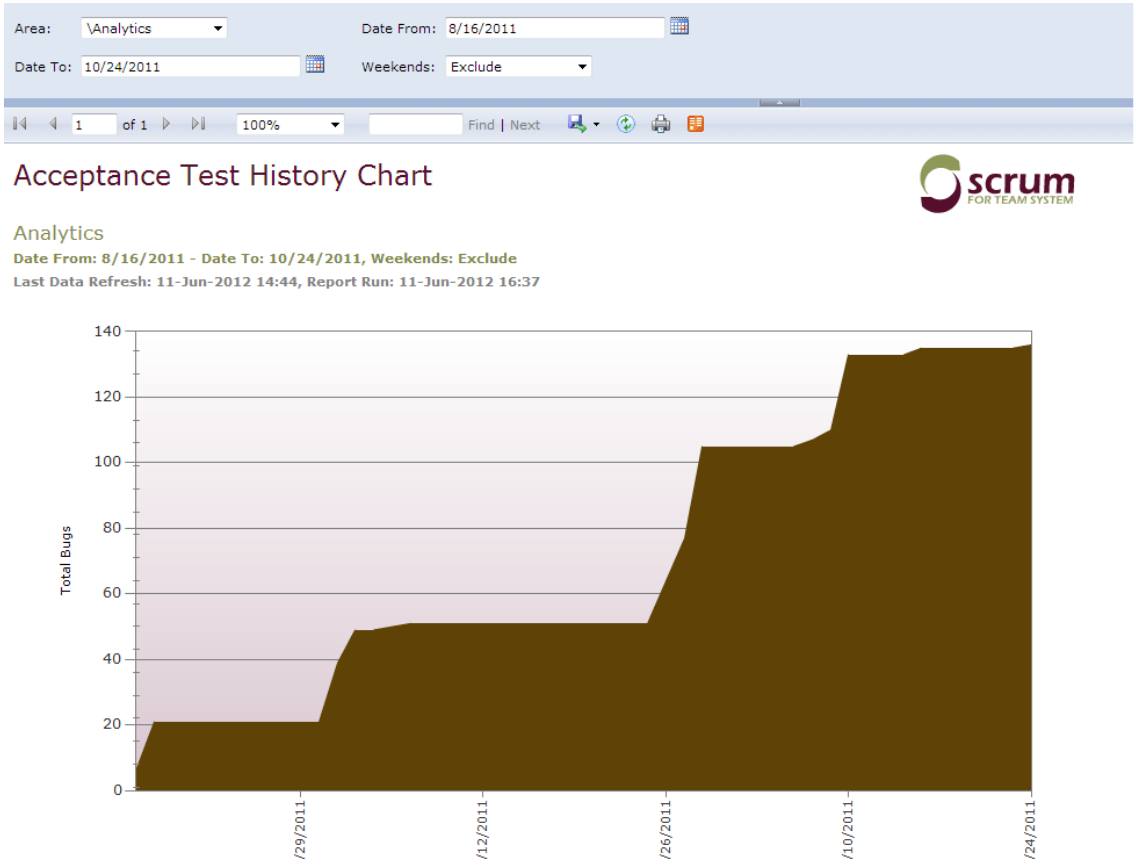

Fig. 6 – Graficul de acceptanta

4.3 Teste de acceptanta

Graficul de teste de acceptanta prezinta evolutia numarului de teste de acceptanta pe parcursul perioadei de dezvoltare. Fiind realizat de asemenea ca si un raport, se vor putea selecta data de inceput si de sfarsit pentru raport precum si prezenta sau excluderea zilelor de weekend. Precum se poate observa si in graficul din figura 6 pe masura ce trecea timpul de dezvoltare al proiectului, numarul de teste de acceptanta crestea. Acest lucru indica prezenta echipei de testare in cadrul echipei si poate fi un indicator al velocitatii membrilor acestei echipe.Pe parcursul celor doua luni de zile selectate pentru raportul curent s-au produs aproximativ 135 de teste de acceptanta, respectiv pentru sprintul 1 s-au realizat 20 de teste, pentru sprintul 2, 25, pentru sprintul 3, 25 etc.

Graficul are forma unor trepte datorita faptului ca odata realizate testele de acceptanta pentru o anumita parte din produs, s-a trecut la rularea acestor teste. Din acest motiv nu exista o dezvoltare continua a testelor de acceptanta.

4.4 Graficul istoriei bug-urilor

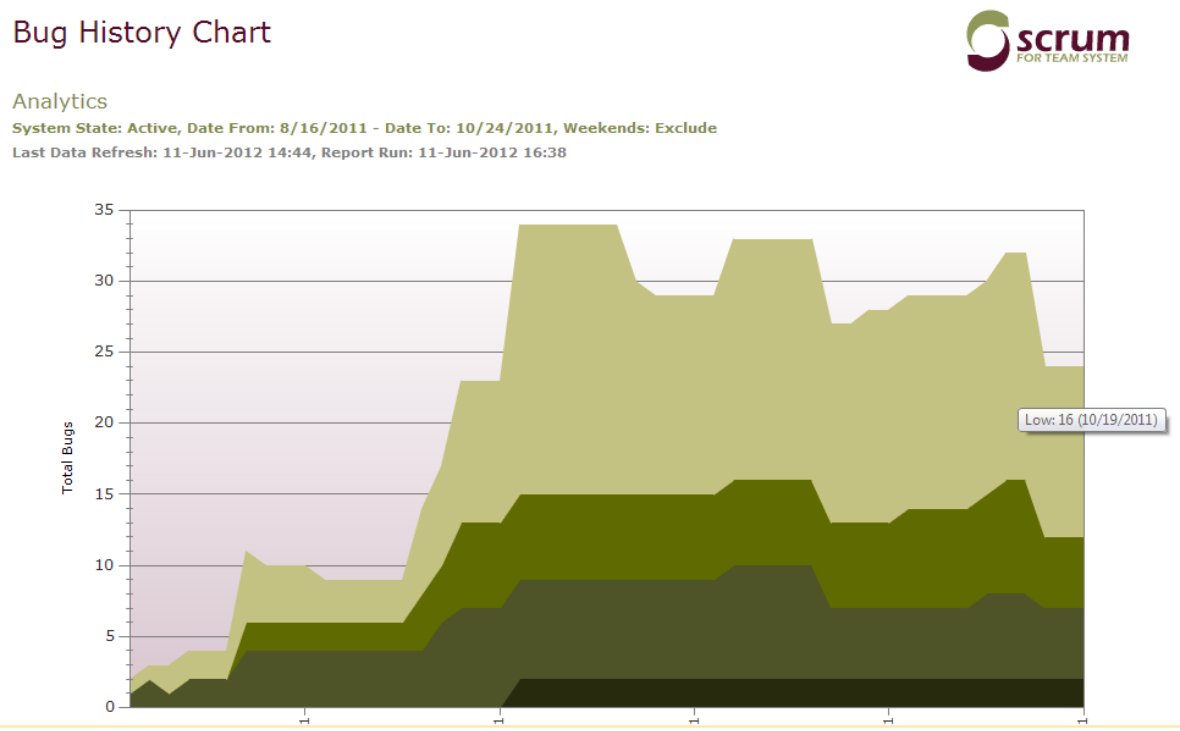

Fig. 7 – Graficul istoriei bug-urilor

Acest grafic ofera nu doar o perspectiva asupra modului de lucru al echipei de dezvoltare ci si al calitatii codului scris de acestia. Pe graficul din imagine se poate observa cresterea numarului de bug-uri pe masura progresarii proiectului. Acest lucru este normal deoarece dezvoltarea unui proiect implica inevitabil existenta unor bug-uri. Se poate observa ca spre deosebire de graficul testelor de acceptanta, acesta are un ritm crescator uneori, si descrescator alteori. Acest lucru se datoreaza faptuli ca in primele sprint-uri s-a realizat doar parte de devellopment fara a fi introdus timp separat pentru rezolvarea eventualelor bug-uri. Pe mesaura trecerii timpului numarul acestora a scazut.

Se pot observa patru culori pe acest grafic. Acestea sunt asociate tipurilor de bug-uri: cu prioritate scazuta, medie, ridicata si critice. Se poate observa ca numarul cel mai ridicat este asociat celor cu prioritate scazuta. Din acest motiv unele dintre ele nu au fost rezolvate.

De pe grafic se poate selecta o anumita perioada pentru care se vor vedea toate bug-urile existente. Acestea sunt sortate in functie de gravitate pentru a putea fi mai vizibile.

5. Concluzii

Managementul proiectelor software a preocupat industria software inca de la inceput. De-a lungul timpului au existat mai multe metode de dezvoltare a proiectelor. In ultimul timp metoda Agile/ SCRUM a avut un trend ascendent fiind in prezent folosita de numeroase companii din lumea intreaga.

Pentru ca un proiect sa poata fi dezvoltat folosind aceasta metodologie trebuie sa existe toate entitatile asociate acestuia: scrum master, product owner, development team (inclusiv echipa de teste). Este important de retinut faptul ca scrum master-ul este cel responsabil ca membrii echipei sa aiba ce lucra, insa nu are drept de decizie. Deciziile privind prioritizarea sarcinilor apartine product owner-ului.

De asemenea pentru ca acest proces sa se paota desfasura corect trebuie sa existe anumite artefacte precum product backlog, sprint backlog, burndown chart. Acestea ajuta echipa la o mai buna organizare, la fel ca si sedintele la care membrii trebuie sa participe: sedinta zilnica, cea de planificare si retrospectiva.

Printre punctele forte ale acestei metodologii aminim o mai buna organizare, chiar si in cazul lipsei specificatiilor, data de finalizare flexibila si adaptabila nevoilor clientului, claritate pentru membrii echipei in ceea ce priveste sarcinile. Singurul punct negativ al acestei metodologii il reprezinta existenta factorului de stres asupra membrilor echipei datorita lipsei momentelor „moarte” in procesul de dezvoltare.

6. Referinte

[1] http://scrumforteamsystem.codeplex.com/

[2] http://www.rallydev.com/

[3] http://extremeprogramming.org/

[4] James Shore, The Art of Agile Development, O'Reilly Media, October 2007

[5] Mike Cohn, Succeeding with Agile, O'Reilly Media, November 2009

[6] http://xprogramming.com

[7] http://agilealliance.org

[8] http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx

[9] http://agileworks.ro

[10] Jez Humble, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley Professional, august 2010