Raport de Dezvoltare
MPS - Milestone 3

Membrii echipei:
Blacioti Mihai - Tester
Brînză Maria-Cristina – Software Developer
Dranca Ștefana-Ioana – Team Leader
Dumitrescu Ioana-Teodora – Project Manager
Lovin Cosmin-Viorel - Software Developer

1. Planul de risc al dezvoltării proiectului

În urma unui brainstorming la nivel de echipă și a unei analize amănunțite a task-ului și implementării alese spre dezvoltare într-un timp restrâns, am identificat o serie de riscuri ce pot îngreuna sau chiar sista producția. Pe unele dintre acestea am reușit să le eradicăm, pe altele să le diminuăm, dar încă există riscuri active asupra cărora acordăm o atenție sporită pentru a nu le lăsa să scape de sub control.

Am abordat în detaliu problemele apărute cu ajutorul planului de risc din link-ul următor: Risk Management Plan MPS.

Am considerat că cele mai multe dificultăți apar din cauza complexității pe care am ales-o raportat la timpul de dezvoltare a aplicației, dar și a dorinței de a atinge o acuratețe ridicată pentru fiecare rulare a algoritmului.

Ajungând la finalul dezvoltării proiectului în timp util și fără probleme majore întâmpinate, considerăm că am abordat riscurile într-o manieră preventivă cu rezultate satisfăcătoare.

2. Succesul echipei în dezvoltarea soluției

IMPLEMENTARE

Până în acest moment am reușit implementarea aplicației software atât pentru binarizarea locală, cât și pentru cea globală. Bucățile de cod problematice au fost retușate sau înlocuite și au avut loc sesiuni de brainstorming pentru a discuta dacă se păstrează abordarea aleasă sau nu (în cazul binarizării globale) și ce modificări apar in cazul celei locale.

S-a întocmit un raport tehnic în care se explică exact ideile de implementare.
Pentru binarizarea globală, soluția propusă are la bază utilizarea unui algoritm de tip Monte Carlo care generează în mod aleator arbori de operații între pragurile algoritmilor dați, pentru ca în final să extragem cei mai buni doi arbori (de pe train și validation), care sunt apoi comparați pe setul de date pentru testare.
Pentru cea locală,
În ambele cazuri s-au implementat și funcțiile în C++ care să traducă implementarea.

Pentru binarizarea locală, programul iterează prin fișierele CSV din care se extrag, pentru fiecare pixel, pragurile obținute de algoritmi, valoarea și clasa sa în imaginea de ground truth (0 sau 1). Pentru a manipula cu ușurință datele am implementat o clasă Pixel, iar fiecare reader salvează într-o listă toți pixelii săi. Apoi, se

aplică în mod repetat următorii pași: pentru fiecare reader se inițializează cu 0 variabilele în care se contorizează numărul de pixeli true positive, false positive, true negative și false negative; se aleg un număr aleator de valori din fiecare listă cu pragurile date pentru fiecare pixel, pentru care se aplică câte o funcție matematică aleatoare, se adaugă în lista cu praguri rezultatul funcției; se încadrează cu ajutorul acestui nou prag pixelul curent într-una dintre categoriile menționate; după parcurgerea tuturor pixelilor dintr-un reader se calculează valoarea F-means pentru iterația curentă (asociată cu un arbore) și se adaugă la scorul total.

TESTARE

Testarea propriu-zisă a funcționalității codului a constat în primul rând în rularea pe secțiuni a programelor, adică verificarea următoarelor:

- Citirea după formatul corect din fișierele de intrare
- Returnarea rezultatelor funcțiilor matematice în conformitate cu variabilele alese
- Reconstrucția funcției care generează cel mai bun scor

În al doilea rând, soluția a fost testată pe fragmente de dimensiuni reduse din input-ul oferit, pentru a asigura că rezultatele pot fi înțelese cu ușurință. Ulterior, ne-am îndreptat treptat spre testarea totală a proiectului, obținând rezultate corespunzătoare comparabile cu cele ale altor echipe.

De menționat este că întregul proiect a respectat standardele impuse în planul de QA realizat de testerul echipei (link: Software Quality Assurance).

3. Impactul metodologiei de dezvoltare folosite

Faptul că ne-am folosit de metodologia de dezvoltare Agile, varianta Scrum ne-a ajutat extrem de mult în asignarea în mod egal din punct de vedere al volumului de muncă și al timpului necesar soluționării task-ului. Astfel, task-urile au fost bine stabilite de la bun început, fiecare având un overview al situației generale a proiectului, dar cunoscând în același timp și statusul curent al celorlalți membri ai echipei. Totuși, am pus accent și pe lucrul în echipă, oferind sau cerând ajutor pentru diverse task-uri.

Funcționalitatea de backlog a platformei Jira pe care o folosim ne-a ajutat să stabilim o cursivitate a etapelor de implementare, să putem împărți timpul total mai eficient prin cunoașterea tuturor task-urilor până la final.

La ședințele de planificare a sprintului am estimat task-urile corespunzătoare acestuia pentru a le atribui membrilor echipei în funcție de productivitatea deja cunoscută a acestora. De asemenea, am învățat să ne adaptam mai bine volumul de muncă per sprint.

Prin retrospectivă, am evaluat propria performanță, am trecut în revista cunoștințele noi dobândite și am oferit feedback pentru a ne afla într-un constant progres organizatoric.

4. Monitorizare, evaluare și controlul evoluției proiectului

MONITORIZARE

Pe lângă întâlnirile de la laborator, am organizat o serie de ședințe de verificare a progresului, unde fiecare a prezentat statusul task-urilor asignate, blocajele întâmpinate și rezoluțiile de viitor. În plus, am încercat să îmbunătățim în mod constant abordarea aleasă pentru soluționarea fiecărei sarcini, acest fapt ducând de obicei la o dezbatere.

Conflictele în cadrul unui proiect sunt inevitabile, însă asta nu înseamnă că blocajele produse de acestea nu pot fi gestionate. De obicei, consensul era obținut ușor, chiar dacă fiecare membru al echipei a avut un motiv anume pentru orice decizie și o justificare pe măsură. În momentele în care discuția nu avansa către o concluzie am recurs la utilizarea tacticilor de rezolvare a conflictelor. Strategia cel mai des folosită a fost cea de negociere, fiind susținută de tot colectivul. Pentru îmbunătățirea funcționalității metodei, am apelat la o lista Pro/Contra, gestionând fiecare item cu o pondere a importanței. Urmărind acest set de reguli, conflictele erau soluționate ușor. Cu toate acestea, în condițiile în care eșuam, se opta mai întâi pentru ocolire, ulterior abordând direct problema, dreptul de veto revenind project managerului.

EVALUARE

Evaluarea performanțelor membrilor echipei s-a realizat de către project manager, întrucât acesta a interacționat cel mai mult cu părțile implicate.

A existat un formular de evaluare atât pentru al doilea milestone al proiectului, discutat în raportul anterior, cât și pentru etapa finală. Acțiunea s-a repetat întocmai pentru a observa eficiența comunicării constante și a feedback-ului primit.

Comparând rezultatele, s-a obținut o creștere vizibilă a performanțelor (link: Raport de dezvoltare - M3).

CONTROLUL EVOLUTIEI

Una dintre metricile pe care le-am folosit pentru evaluarea eficienței proiectului în fază intermediară este *SPI (Schedule Performance Index)*:

EV (“Cât de mult s-a realizat până în acest punct din ce s-a planificat?”)

Dacă la milestone 2, indexul ne arăta că suntem ușor în întârziere (0.9), în acest moment proiectul este finalizat în procent de 100%. Ca în fiecare caz, există loc de îmbunătățire a scorului și timpilor de rulare, însă echipa este mulțumită de performanța curentă.