NEGRILA ANGELICA
441 A

MODELE DE DEZVOLTARE SW SUPLA, RAPIDA

Descriere generala

Prototiparea software reprezinta o etapa importanta din cadrul unui proiect in care se defineste o arhitectura conceptuala a aplicatiei care va reprezenta baza procesului de implementare. Intregul proces de dezvoltare al unui sistem se bazeaza pe interactiunea cu clientul/utilizatorul, iar prototipul reprezinta punctul central al analizelor si discutiilor care duc la definitivarea arhitecturii.

Prototipul poate fi considerat ca o colectie de specificatii formale care descriu la nivel superior si mijlociu structura unei solutii software. O definitie simpla descrie prototipul ca fiind schema unei aplicatii software.

Etapa de implementare (dezvoltare) reprezinta redactarea codului aplicatiei si testarea permanenta a acestuia pentru a fi in conformitate cu specificatiile stabilite. Trebuie subliniat faptul ca prototipul este redactat inaintea etapei de implementare, dar poate fi modificat permanent in functie de analizele tehnice realizate si discutiile purtate cu beneficiarul. De altfel, interactiunea dintre dezvoltator si client trebuie specificata si formalizata prin intermediul unor documente in cadrul procesului de dezvoltare. Acesta reprezinta o colectie de proceduri prin care se reglementeaza etapele de proiectare, dezvoltare, testare si predare ale unui proiect software, precum si interactiunile dintre partile implicate (echipele companiei care realizeaza implementarea: dezvoltatori, testeri, arhitecti, ingineri de sistem, manageri, etc. si client). Au fost definite doua abordari referitoare la prototipuri:

- throwaway : prototipuri care sunt defintе rapid pentru a fi redefinite intr-o iteratie ulterioara
- evolutionary: prototipuri care sunt construite pentru a fi modifcate partial in decursul iteratiilor, insa structura de baza trebuie sa ramana constanta.

Beneficii

Beneficiile dezvoltarii un prototip mai nou în procesul de software-ul sunt:

- neînțelegerile dintre dezvoltatorii de software și utilizatori pot fi identificate de funcțiile sistemului
- serviciile de utilizator lipsă pot fi detectate.
- dificil de utilizat sau servicii confuze ale utilizatorilor pot fi identificate
- personalul de dezvoltare software-ul poate găsi cerințe incomplete și / sau inconsecvente ca prototip dezvoltat.
- deși limitat, sistemul este disponibil pentru a demonstra fezabilitatea rapida și utilitatea punerii în aplicare a managementului.
- prototipul serces ca o bază pentru scrierea “caietului de sarcini” pentru producerea unui sistem de calitate.

Prototipurile software au, de asemenea si alte utilizări:

- Formarea utilizatorului
- Sistemul de testare

Etapa finala a procesului de evaluare este de prototip in evolutie. Momentan nu sunt probleme managerial non-tehnice care pot determina vreo dificultatea de a utiliza prototipuri în unele organizatii.

Tehnici de prototipare

In paragrafele urmatoare vor fi prezentate mai multe tehnici de prototipare utilizate, avand caracteristici specifice.

Atunci cand un proiect este dezvoltate prin prototipare, sunt remarcate o serie de avantaje:

- Reducerea timpului de dezvoltare si a costurilor implicate: Prototiparea presupunea realizarea unei analize specializate, cu ajutorul careia se pot identifica riscurile si se pot evita costuri aditionale nedorite de dezvoltare. In plus, se poate realiza o impartire a sarcinilor de lucru mai simpla intre membrii echipei, stabilind interfete corespunzatoare.
- Interactiunea mai buna cu clientul si membrii echipei de dezvoltare: Clientul poate oferi un feedback rapid pentru a evita neintelegerile aparute dupa demararea procesului de implementare. De asemenea, si membrii echipelor participante in cadrul proiectului pot transmite pareri referitoare la prototipul lansat, permitand integrarea acestora intr-o forma viitoare.
- Eficientizarea divizarii sarcinilor de lucru si imbunatatirea lucrului in echipa: Odata ce prototipul este definit, pot fi redactate si documentate task-urile necesare, pe care apoi managerii le pot transmite angajatilor. Respectarea interfetelor stabilite si implementarile corecte si conforme specificatiilor pot reduce timpul de dezvoltare.

Desigur, utilizarea prototipurilor aduce cu sine si o serie de dezavantaje:

- Analiza incorecta duce la pierderi semnificative: In cazul in care analiza realizata de arhitectii de sistem nu surprinde in mod corect detaliile esentiale ale proiectului, prototipul realizat poate sa nu fie util, iar procesul de re-design (sau intr-un caz mai nefericit de re-implementare) costa.
- Neintelegerea prototipului: Daca specificatiile redactate nu sunt clare, pot aparea discutii confuze intre client si dezvoltator. De asemenea, se poate intampla ca membrii echipei sa dezvolte cod neconform cu specificatiile reale.
- Dezvoltarea excesiva a prototipului. Aceasta problema apare atunci cand arhitectii nu reusesc sa finalizeze prototipul in timp util din cauza diverselor probleme (neintelegi cu clientul, divergente in cadrul echipei, etc.). In aceasta situatie se ajunge sa se investeasca prea mult in etapa de prototipare si se produc intarzieri in cadrul procesului de dezvoltare.

In decursul timpului au fost identificate mai multe metode de dezvoltare cu ajutorul prototiparii. O data cu proliferarea dezvoltarii agile si cresterea importantei lucrului in echipa, rolul prototipului devenit si mai relevant. In continuare vom prezenta cateva metode de dezvoltare bazate pe prototipare.

Dynamic Systems Development Method (DSDM) este o metoda certificata ISO 9001 pentru a dezvota solutii, avand la baza prototipare. Presupune iteratii in 4 pasi:

1. Identificarea necesitatilor prototipului
2. Stabilirea unui plan de dezvoltare
3. Dezvoltarea prototipului
4. Revizuirea prototipului

Prototiparea operationala presupune imbinarea prototipurilor throwaway cu cele evolutionary. Componentele bine cunoscute sunt dezvoltate ca prototipuri evolutionare, in timp ce legaturile dintre acestea sunt definite in cadrul unor prototipuri throwaway. Se recomanda initierea interactiunilor cu clientul prin implicarea unui arhitect in discutiile cu acesta, arhitect care va avea rolul de a redesena prototipurile throwaway. Aceasta metode imbina dezvoltarea standard cu utilizarea unor trucuri considerate quick-and-dirty pentru rezolvarea problemelor pe care echipa de dezvoltare nu cunoaste cum sa le solutioneze. Ideea din spate ii apartine lui Alan Davis.

O metoda mai rigida de dezvoltare este reprezentata de **Evolutionary Systems Development (ESD).** Aceasta urmareste definirea unui prototip de baza ca punct central de dezvoltare care sa sufere mici modificari intre iteratiile de implementare. O particularizare mai agila a acestei metode numite Systemscraft a fost descrisa de John Crinnion si a fost prezentata in cartea sa " Evolutionary Systems Development". Procedurile presupun o serie de analize clasice standard, iar rezultatele sunt considerate ca fiind de cele mai multe ori bune.

Evolutionary Rapid Development (ERD) este una din cele mai folosite tehnici in momentul de fata. A fost documentata de Software Productivity Consortium, organizatie aflata in subordinea DARPA. Una din cele mai importante caracteristici este date de reutilizarea solutiilor dezvoltate. De cele mai multe ori se lucreaza in echipe mici care dezvolta colectii de solutii pentru anumite obiective, in timp ce alte echipe lucreaza pentru integrarea acestora. In plus, elementul cheie este clientul. Se recomanda intalniri frecvente cu clientul pentru a prezenta acestuia stadiul produsului, pentru a verifica acceptarea solutiilor propuse de catre acesta, dar si pentru a propune integrarea de tehnologii noi care nu erau disponibile la un moment anterior. Astfel, se asigura o flexibilitate foarte buna si se livreaza calitatea asteptata de catre client la momentul predarii.

Este comuna utilizarea unor iteratii mici numite timeboxes. In cadrul acestora se asigura implementarea unui set redus de functionalitati care sunt testate si prezentate clientului in cadrul intalnirilor stabilite.

Interfata cu utilizatorii

Sistemele de generare a interfeței se pot baza pe interfata de management de sistem care oferă funcțiile de bază ale interfeței, cum ar fi selectarea meniului, afișare obiect și așa mai departe. Din punct de vedere al ingineriei software, este important să realizăm că prototipurile interfață cu utilizatorul sunt o parte esențială a procesului.

Implicarea clientului in software scade si implicarea contractantului creste cu cat soecificatiile se dezvolta. Detectarea erorilor este argumentul cel mai puternic pentru dezvoltarea unei specificații formale.

Prototipurile de interfață sunt ecrane detaliate aproape similare "look and feel" ca o soluție finală.Scopul este de a evidenția punctele forte și punctele slabe ale utilizabilității interfatei si esteticii. Într-o aplicatie GUI (Graphical User Interface) ,se poate considera că această formă a prototipurilor sa

includa fonturi, scheme de culori, convenții vizuale, stiluri de interacțiune (de exemplu, un singur click, dublu click, răsturnare, scurtaturi, etc) , utilizarea de caracteristici audio (de exemplu, faceți clic pe sunete, sunete de avertizare, etc), precum și accesibilitatea pentru fiecare stoc de ecrane. Pot exista mai multe variante de produse.

Unelte si proiecte

Pentru implementarea unui proiect prin metode de prototipare sunt necesare o serie de aplicatii software care imbunatatesc productivitatea echipei angajate.

In primul rand, se recomanda utilizarea aplicatiilor pentru managementul cerintelor intr-o maniera cooperativa, care sa permita accesul clientului la vizualizarea acestora si sa poate solicita schimbarile pe care le considera necesare. Un exemplu in acest sens este reprezentat de Rational Doors, de la IBM.

Pot fi utilizate aplicatii grafice pentru a defini si prezenta arhitectura proiectului. Spre exemplu, se pot utiliza aplicatii precum Rational Rose sau Microsoft Visio pentru a construi structura software a proiectului. De cele mai multe ori, aceste aplicatii ofera si suport pentru generarea claselor in diverse limbaje de programare (C++, Java, .NET, etc.). Astfel, cei care scriu efectiv codul aplicatiei au sarcina de a completa clasele deja definite pentru a asigura functionalitatea completa a sistemului. In paralel se pot construi module speciale pentru testarea claselor definite, module care isi au originea tot in schema software definita de catre arhitecti.

O data ce prototipul este definit, se procedeaza la implementare codului, derulata in paralel cu simularea si testarea segmentelor functionale. IBM Functional Tester este o aplicatie specializata in operatiuni de testare care poate rula atat teste in linie de comanda, cat si teste automatizate pentru interfata grafica.

In general, proiectele bazate pe prototipare folosesc aplicatii specializate pentru a realiza etapele specifice de analiza, implementare, testare si remediere a problemelor aparute. De asemenea, se recomanda efectuarea testelor intr-un mediu specializat (de multe ori chiar virtual), pentru e eficientiza utilizarea resurselor si pentru a nu perturba alte componente aflate in productie.

Tratare exploratorie și experimentală

Se remarca doua abordari de testare utilizate. Prima dintre ele se numeste testare experimentala si presupune definirea unui test case, efectuarea testului propriu zis si verificarea rezultatului. In cazul in care rezultatele nu sunt conforme cu ceea ce este specificat in test case in rubrica de rezultate asteptate, tester-ul completeaza un raport prin care se raporteaza defectul echipei de dezvoltare. Metoda este foarte utila pentru a asigura functionalitatile de baza negociate cu clientul.

Cea de a doua metoda de testare este testarea exploratorie. Aceasta reprezinta de cele mai multe ori rularea unor teste “free style” pe care le executa un tester fara a avea un plan stabilit. Nu este necesara elaborarea unui test case, ci se urmareste tocmai testarea aplicatiei in situatii speciale (de ex. Cum reactioneaza un site web daca intr-un camp de adresa IP un user scrie numai caracterul virgula).

Scopul acestei metode este acela de a imbunatati calitatea generala a produsului si de a invata mai multe despre acesta. De asemenea, poate fi imbinata si cu rezolvarea defectelor descoperite pe parcursul testelor, ceea ca mareste eficienta etapei de lucru. De cele mai multe ori, utilitatea testarii exploratorii depinde de calitatile tester-ului care o efectueaza. Un angajat bine pregatit va fi capabil sa inventeze situatii mai diverse si sa observer bug-uri mai greu de depistat. Un dezavantaj al acestei metode este dat de faptul ca, de cele mai multe ori, procedurile se executa manual si nu pot fi automatizate, asa cum se poate face in cadrul testarii experimentale.

Bibliografie :

1. http://woorisol.kyungpook.ac.kr/lab/prof/SoftEng/ch8.htm
2. http://en.wikipedia.org/wiki/Software_prototyping
3. http://www.sqa.org.uk/e-learning/IMAuthoring01CD/index.htm