Evaluarea sistemelor software

11.1. Set de metrici software pentru conducerea proceselor de mentenanţă a programelor

Problema evaluării sistemelor software a devenit cu atât mai acută şi importantă cu cât dimensiunea, costul şi locul strategic ocupat de sistem a crescut în timp.

Aşa cum fiecărui produs i se ataşează la livrarea către beneficiar un certificat de calitate, tot aşa sistemele software, indiferent de tehnica de realizare (clasică sau din domeniul inteligenţei artificiale), trebuie să fie garantate şi studiate din punctul de vedere al fiabilităţii, calităţii şi întreţinerii lor.

Un studiu de specialitate (Stark, 1994) face cunoscute rezultatele cercetărilor unui grup însărcinat cu administrarea programelor din cadrul NASA, care a definit şi implementat un set de metrici software conţinând 13 metrici destinate să corecteze şi să adapteze acţiunile de mentenanţă.

Echipa numită MOD a răspuns de astfel de activităţi de mentenanţă software la diferite niveluri (sisteme, subsisteme, module) utilizând paradigma solicitare (întrebare) metrică, care identifică o mulţime de 13 metrici pentru utilizarea curentă şi de viitor în studiul sistemelor.

Aceste metrici permit evaluarea stării curente a activităţilor de mentenanţă şi predicţionează rezultate viitoare (solicitări) cum ar fi :

-"Cât de mare trebuie să fie echipa de care este nevoie pentru evaluare? "

-"Cât timp va dura lucrul pentru încheierea raportului software?"

Definirea setului de metrici

Literatura de specialitate în mentenanţă software precizează metricile pentru astfel de activităţi.

De asemenea, în lucrarea lui Gill (1991) se pune problema datelor care trebuie păstrate în timpul mentenanţei sistemului, sau care determină activităţi de mentenanţă (Zuse 1992).

Pornind de la acestea, se pot sintetiza problemele legate de activităţile de mentenanţă a software-ului inteligent ca evoluând pe direcţia "cerere- întrebare- metrici" (Tabelul 11.1).

Tabelul 11.1 **Concluziile privind paradigma cerere/întrebare/metrică**

Cerere	Întrebare	Metrici
Maximizarea satisfacţiei clientului	Câte probleme afectează utilizatorul (clientul)?	- Raportul discrepanţelor (DR) şi solicitările de service (SR) de-a lungul desfăşurării. - Fiabilitate software -Raport întrerupere/ funcţionare
	Cât timp durează pentru a definitiva o problemă?	- Terminarea DR/SR - DR/SR de-a lungul desfăşurării.
	Unde sunt "îngustările"	- Utilizarea staff-ului - Utilizarea resurselor calculatorului
Minimizarea efortului şi proiectului	Unde se consumă resursele?	- Utilizarea staff-ului - Planificarea SR - Instanţieri gen ADA - Distribuţia tipului de erori
	Cât de mentenabil este sistemul?	- Utilizarea resurselor calculatorului - Dimensiunea software-ului - Densitatea erorilor - Volatilitatea soft-ului - Complexitatea software-ului
Minimizarea defectelor	Este software-ul susţinut (confirmat) efectiv inginereşte?	- Densitatea erorilor - Raport întrerupere/ funcţionare - Instanţieri ADA - Fiabilitate software

Observaţii:

Metricile sunt utilizabile individual, dar cel mai mare beneficiu derivă din faptul că ele sunt folosite ca o mulţime de paşi în solicitări de concurenţă cum ar fi calitate faţă de productivitate sau calitatea în raport cu datele realizate, etc.

Pentru a obţine aceste beneficii se raportează mai mult de o metrică la fiecare cerere, iar alte metrici întreţin desfăşurarea cererii (de exemplu fiabilitatea software).

Aceste desfăşurări pot furniza nu numai observaţii din interiorul proiectului, dar permit de asemenea verificări asupra consistenţei datelor.

Şefii proiectului pot să combine metricile pentru a investiga paşi nevizibili numai cu o singură metrică.

1.Dimensiunea software-ului

Mărimea, dimensiunea software-ului este un parametru primar de intrare în mai multe modele de estimare a costului şi calităţii software-ului şi este strâns legată de alte metrici din mulţime (cum ar fi densitatea erorilor, complexitate, etc).

Metrica este definită ca numărul de linii de cod sursă (NLCS) care au fost întreţinute în proiect. Aceasta poate fi utilizată în proiectul de management pentru următoarele activităţi:

Mulţimea de metrici pentru mentenanţa software-ului

1a). Urmărirea efortului necesar pentru întreţinerea codului.

- O creştere în dimensiune a software-ului poate conduce la menţinerea unui personal mai numeros de întreţinere.

- Dacă se poate stabili o relaţie între specificaţiile proiectului, prin utilizarea unui model de cost sau a unei regresii liniare simple, atunci mărimea privind schimbarea staff-ului de mentenanţă poate fi predicţionată prin schimbarea în dimensiune a software-ului.

1b). Anticiparea problemele de performanţă în hard, în timp real.

- O tendinţă către creşterea dimensiunii software-ului ar iniţia acţiuni corective care, fiecare, ar contoriza sau s-ar armoniza cu efortul crescut de depanare şi/sau puterea calculatorului.

2. Echipa de software

Schimbările în echipa de software au un impact direct asupra costurilor proiectului şi asupra progresului în mentenanţa software.

Această metrică este reprezentată de numărul de ore de-a lungul unei luni consumate de inginerii software direct implicaţi în activităţile de mentenanţă şi furnizează informaţii de management legate de datele care ar fi necesare pentru o predicţie a viitoarelor cereri ale staff-ului proiectului.

De asemenea, poate fi folosită pentru următoarele acţiuni:

2a). Să estimeze eficacitatea în activităţile de inginerie software prin evaluarea numărului de paşi şi a resurselor necesare în fiecare proces de mentenanţă.

- Este posibil să se continue sau să se elimine paşi din proces, astfel încât să-l facă mai eficient şi să răspundă mai bine utilizatorilor.

2b). Să identifice cele mai intensive resurse din activităţile de mentenaţă (de exemplu, solicitări de schimbare, evaluare, planificare, implementare, test, eliberare de resurse).

2c). Să identifice cele mai intensive resurse din sistem.

Această metrică permite o "verificare de sănătate" a proiectului prin comparaţii cu regulile de acţiune recunoscute în industrie pentru cheltuiala cu personalul de-a lungul fazei de mentenanţă, din ciclul de viaţă al sistemului software.

3. Procesarea solicitării de mentenanţă

Această metrică monitorizează fluxul activităţii de mentenanţă software şi determină nivelul de satisfacere a clientului.

Maximizând satisfacţia utilizatorului, este important să se manipuleze solicitările cele mai vizibile pentru client, asigurându-i astfel o asistenţă permanentă.

Metrica permite şi unui analist să execute următoarele funcţii:

3a). Să facă o predicţie asupra cantităţii de muncă necesară datorită noilor solicitări, sau noilor rapoarte, comparându-le cu cereri similare din alte proiecte.

3b). Să determine nivelul satisfacerii clientului de către produs, conducând cu grijă evoluţia în echipa de management.

3c). Să efectueze studii de ocupare a personalului, a resurselor calculatorului, reducând posibilele rămăşiţe din modelul de simulare a evenimentelor discrete ale proceselor în cauză.

Această metrică reprezintă o bună cale de a observa tendinţa cumulată a muncii rămase de efectuat, sau pentru a estima cantitatea totală de muncă solicitată. În mod ideal, "progresul" ar trebui să fie aproape zero.

4. Planificarea mărită a software-ului

Metrica de planificare sporită a software-ului trasează mărimea timpului necesar pentru includerea unei cereri sporite şi efortul ingineresc consumat pentru această solicitare.

Sunt folosite două date primare pentru a calcula această metrică:

1). Numărul planificat şi actual de ore de inginerie cheltuite cu acestă suprasolicitare;

2). Timpul calendaristic scurs de la apariţia cererii până la rezolvarea ei, ca facilitate de utilizare.

Mulţimea de metrici include şi această metrică deoarece permite managerilor proiectului să identifice producţiile de plan cu cel mai înalt risc, să evalueze timpul necesar, pentru a oferi utilizatorilor o cerere sporită şi să predicţioneze cantitatea de muncă pe care o incumbă această cerere.

De exemplu, când se primeşte o cerere de service i se asignează o prioritate (aceasta semnifică risc pentru plan, sistem, proiect), datele de care are nevoie şi se estimează personalul necesar pentru satisfacerea completă a task-ului.

Trasând aceste estimări peste actuala desfăşurare de date a solicitării adiţionale, conducerea proiectului poate să-şi modifice procesul estimat şi să furnizeze utilizatorilor o informaţie mai bună în ceea ce priveşte schimbările efectuate în sistem.

5. Utilizarea resurselor calculatorului

Metrica privind utilizarea resurselor calculatorului (CRU) marchează modul de utilizare a resurselor sistemului.

Sunt incluse patru resurse, şi anume: CPU, disc, memorie şi utilizarea canalului I/O. Metricile CRU sunt raportate ca procentaje din capacitatea resurselor şi furnizează un cadru pentru prezentarea concluziilor analizei rezultatelor sistemului în raport cu contractul (Kern, 1992).

De asemenea, avertizează mai devreme conducerea proiectului în cazul în care utilizatorii s-au apropiat de limitele de capacitate ale resurselor sistemului.

Previziunea că volumul de date memorate se apropie de capacitatea de 90% se potriveşte cu o dreaptă de regresie de forma:

capacitate = *rata_de_ocupare* * *luna* + *constantă*, de la punctele datei până la ultima acţiune corectivă.

Această linie intersectează pragul de 90% capacitate în octombrie 1993, când a fost efectuată ultima acţiune corectivă.

6. Densitatea erorilor

Numărul rapoartelor de discrepanţe incluse într-un proiect cu un software fix, per KNLCS, în funcţie de timp, defineşte metrica de densitate sau frecvenţa erorilor. Densitatea erorilor măsoară calitatea codificării şi poate fi utilizată pentru:

6a). Predicţia numărului de erori remanente în cod prin utlizarea unui model de calitate;

6b). Stabilirea densităţii standard de erori în scop de comparaţie şi predicţie pentru fiecare nivel sever de defectare sau tip de eroare.

- Aceasta permite proiectanţilor să identifice sistemele sau procesele cărora trebuie să li se acorde o atenţie deosebită.

7. Volatilitatea software-ului

Belady şi Lehman au definit în 1976, pentru prima dată "volatilitatea software-ului".

Este un raport dintre numărul de module schimbate din cauza cererii de mentenanţă şi numărul total de module eliberate în timp.

Sarcina acestei metrici este de a măsura cantitatea de schimbări în structura software-ului, în timp.

Utilizată împreună cu fiabilitatea, CRU şi complexitatea proiectată, ea ajută managerii să decidă dacă o procedură calificată de test ar putea fi reexecutată şi să identifice nevoia de reconstruire a software-ului.

Se pot stabili două reguli de bază pentru această metrică, şi anume:

(1) Când volatilitatea depăşeşte 30% este necesară o recalificare pentru o utilizare operaţională;

(2) Dacă metrica depăşeşte 80% pentru produsul realizat, sistemul va trebui reconstituit.

8. Durata raportului de discrepanţe

Această metrică (DR) oferă o măsură a timpului necesar pentru rezolvarea tuturor rapoartelor de discrepanţe din momentul descoperirii lor şi până la încheierea proiectului.

Durata unei discrepanţe este calculată din punct de vedere al datelor raportate, minus datele supuse acţiunii şi furnizează o privire din interior a eficienţei procesului de depanare.

O mărime semnificativă a vechilor DR-uri poate indica că au fost necesare mai multe resurse pentru a le închide, a le lichida.

Examinarea numărului şi vârstei DR-urilor cu prioritate critică ridicată permite managerului de proiect să estimeze timpul de răspuns în depanare.

De asemenea, un număr mic de DR-uri poate indica existenţa unor probleme dificile care necesită schimbări extensive pentru corecţie, sau probleme care nu au fost bine înţelese şi solicită mai multă atenţie.

În plus, managerii responsabili cu procesele de discrepanţe păstrează data şi durata vechilor rapoarte atunci când primesc altele noi.

Suma acestor "vârste" şi ciclurile individuale de timp sunt utilizate pentru a identifica ”îngustările” din domeniu şi a îmbunătăţi procesul.

9. Raportul întrerupere/funcţionare

În lucrările de specialitate s-a comunicat că o schimbare în software are numai 20% şanse de succes dacă implică mai mult sau chiar 50 NLCS şi aproape 50% şanse de modificări făcute corect, pentru mai puţin de 10 NLCS într-o primă încercare.

Raportul întrerupere/funcţionare este numărul de erori inserate în soft-ul operaţional, în liniile de bază, împărţit la numărul de modificări făcute în produsul software.

De exemplu, dacă sunt trei discrepanţe care trebuie corectate şi în urma corecţiei rezultă o eroare, raportul va fi 0,33.

Aceasta înseamnă că activitatea de corecţie a avut eficacitate 67% în rezolvarea DR-urilor. Metrica ajută de asemenea la execuţia următoarelor funcţii:

a). Estimarea numărului de probleme care afectează clientul sau a numărului de erori remanente în cod, utilizând un model de simulare;

b). Identificarea sursele care pun probleme pentru software, ca şi dezvoltarea şi aprobarea lui.

Aceasta este, de asemenea, necesară conducerii pentru urmărirea inspecţiilor care privesc codul şi testarea proiectului.

Metrica raportului măsoară eficacitatea organizării mentenabilităţii în ceea ce priveşte modificările în software-ul operaţional de bază.

10. Fiabilitatea software-ului

Fiabilitatea software-ului poate fi definită drept probabilitatea ca software-ul să nu fie defect într-o perioadă specificată de timp şi în condiţii, de asemenea specificate.

Bazându-se pe datele operaţionale de defectare, metrica "fiabilitatea software-ului" furnizează o indicaţie a numărului de erori de-a lungul unei perioade de durată dată.

Ea trasează rata erorii software-ului curent prin date.

Metrica poate fi utilizată să controleze traficul schimbărilor prin sistem astfel încât să fie menţinută o rată acceptabilă a erorilor.

Dacă sistemul este realizat bine în ceea ce priveşte o "mulţime prag" de mentenabilitate, atunci vor fi permise schimbări complexe.

Aproape toate proiectele au o cerinţă legată de fiabilitatea software, deoarece există un punct de la care rata erorilor din software face sistemul să fie inutilizabil.

11. Complexitatea proiectului

Complexitatea proiectului, ca metrică, indică numărul de module cu care complexitatea proiectului a crescut faţă de ceea ce s-a stabilit iniţial.

În acest caz se utilizează metrica complexităţii extinse a lui McCabe, care contorizează numărul de căi de execuţie independente de-a lungul unei piese date de software, ceea ce înseamnă de fapt numărul de ramificaţii logice plus 1.

De asemenea, ea permite conducerii proiectului să schiţeze posibilităţile furnizorului de a menţine un nivel acceptabil al complexităţii, la nivelul fiecărui modul în parte.

Complexitatea este strâns corelată cu efortul de mentenanţă al programatorului şi cu numărul de erori găsite de-a lungul testării şi operării.

12. Distribuţia tipului de erori

Această metrică prezintă raportul desfăşurat de discrepanţe în trei moduri:

(1). Prin desfăşurarea codului (aceasta înseamnă: hardware, software, resurse umane, neputinţă de a fi duplicat ş.a.m.d.);

(2). Prin tipul problemei care a fost greşită (adică logic, computaţional, interfaţă, data de intrare, etc.)

(3). Prin procesul care a introdus problema (adică cereri, proiect, cod, test).

Toate acestea servesc la identificarea acelor aspecte ale procesului care sunt "susceptibile" la erori, precum şi celor mai comune tipuri de erori introduse.

Informaţia provenind de la această metrică trimite înapoi în lanţul de dezvoltare şi conducere, astfel încât managerii pot să ia efectiv măsuri de reducere a riscului (de exemplu o mai bună inspecţie) şi de asemenea să reducă tipul şi cauzele erorilor.

De asemenea, metrica este utilizată pentru identificarea solicitărilor pentru instrumente de mentenanţă software mult mai utile.

Tabelul 11.2 arată comportarea unui proiect MOD.

Tabelul 11.2 **Rapoarte de probleme care au apărut la un proiect MOD**

Desfăşurarea codului	Rapoarte de probleme
Incapabil de a reproduce problema	317
Problema de duplicare	185
Configurare sau limite de proiectare	95
Software fix	265
Erori umane	28
Hardware fix	5
Altele	77

13. Instanţieri tip ADA (limbajul de programare ADA)

Metrica "instanţieri tip ADA" reprezintă un număr de unităţi generice ADA şi codificate de-a lungul activităţii de mentenanţă, dimensiunea unei unităţi generice (în NLCS) şi o mărime a timpului în care unitatea generică este instanţiată.

Cerinţa acestei metrici este de a marca numărul şi dimensiunea componentelor reutilizabile dezvoltate de proiect pentru folosirea lor în interiorul proiectului sau în alte proiecte.

Această metrică marchează unităţile generice ADA reutilizate şi poate fi utilizată atunci când sistemele de inteligenţă artificială folosesc drept "cunoştinţe" de profunzime algoritmi precompilaţi.

11.2. Programul de implementare a metricilor

După definirea mulţimii de metrici trebuie să se treacă la colectarea, analiza, raportarea şi utilizarea metricilor.

Primul pas în implementarea lor este de a emite o directivă de management. Acest pas este considerat pozitiv deoarece asigură consistenţa de-a lungul proiectului şi demonstrează că cel mai înalt nivel al proiectului, suportă efortul.

Importanţa acestui suport nu poate fi exagerată; fără el metricile programului nu ar fi luate în serios, în particular de contractanţi.

Al doilea pas este de a furniza instrumente automate pentru a susţine metricile de program. Se utilizează spreadsheet-uri pentru a rezuma şi raporta metricile datelor.

Observaţii:

Două dintre metricile de bază, şi anume complexitatea proiectului şi fiabilitatea software, sunt dificil de evaluat şi calculat manual, şi de aceea necesită instrumente automate (chiar gen sisteme expert) care să conducă competent astfel de activităţi..

Echipa MOD recomandă ca instrument "Modelarea statistică şi estimarea funcţiilor de fiabilitate pentru software" (SMERFS) pentru măsurarea fiabilităţii software.

Acest instrument include o varietate de domenii incluzând IBM-PC -urile şi calculatoarele compatibile IBM-PC.

Instrumentul de măsurare al complexităţii este UX-metric / PC-metric de la SET Laboratories (1990).

Acesta este un instrument comercial ieftin care contorizează NLCS, comentarii, linii albe şi calculează metricile de complexitate pentru o varietate de limbaje de generaţia a treia incluzând FORTRAN, C şi ADA.

El lucrează pe staţii şi suportă sistemul de operare UNIX la fel de bine ca şi pe PC-uri, IBM compatibile.

O astfel de activitate trebuie organizată sistematic, începând cu proiectarea, achiziţia de cunoştinţe şi continuând cu implementarea, testarea şi mentenanţă sistemelor de programe pentru a putea "garanta" calitatea produsului software elaborat şi a-i demonstra, şi în această manieră, superioritatea faţă de programele tradiţionale.