FIABILITATEA *M*-APLICAȚIILOR BAZATE PE TRANZACȚII

dr. Paul Pocatilu
Prof. dr. Ion Ivan
Asist. Marius Popa
Asist. Cristian Toma
dr. Lukacs Breda

Facultatea de Cibernetică, Statistică şi Informatică Economică,

Academia de Studii Economice, Bucureşti

Rezumat: Sunt prezentate principalele aspecte privind aplicațiile mobile şi calculul mobil. Totodată, se evidențiază funcțiile pe care trebuie să le prezinte acest gen de sisteme. Sunt prezentate principalele caracteristici ale tehnologiilor wireless existente, care se constituie în suport pentru m-aplicații, modele de estimare a fiabilității software, precum şi aplicația realizată în scopul estimării fiabilității software a m-aplicațiilor.

Cuvinte cheie: aplicații mobile, calcul mobil, tranzacții, fiabilitate, securitate.

1. Introducere

Tehnologii precum „aplicațiile mobile" şi „calculul mobil" (mobile application, mobile computing – *m*-application, *m*-computing) au făcut suficiente progrese pentru a integra comunicațiile şi procesarea datelor în aplicații de tip *e*-commerce (comerț electronic), şi, mai concret, de tip *m*-commerce. *M*-aplicațiile reprezintă o nouă piață care lasă companiilor posibilitatea de a oferi clienților servicii ce presupun accesul la informație şi efectuarea de tranzacții complexe cu telefoane mobile şi cu dispozitive de dimensiune redusă.

O dată cu apariția produselor hardware şi software pentru *m*-aplicații, bazate pe comunicații şi Internet, din ce în ce mai performante şi mai ieftine, se aşteaptă o creştere consistentă a *m*-commerce-ului, denumit şi wireless commerce. Din perspectiva tehnică, *m*-application şi *m*-computing permit integrarea tehnologiilor care oferă suport pentru multimedia, decizii şi tranzacții interactive şi instantanee.

De exemplu, un sistem telemedical este un proces automat, fiind construit cu ajutorul unui agent software, care alertează spitalul dacă starea de sănătate a pacientului se înrăutățeşte. Agentul software preia informațiile de la o *m*-aplicație de tip midlet. O tehnologie de agenți mobili a fost dezvoltată de IBM în 1997 sub denumirea de Aglets.

Simultan, un alt produs software este declanşat de agent pentru realizarea unei videoconferințe între pacient şi furnizorul de servicii medicale. Totodată, un motor de căutare este activat, în timp real, pentru a ajuta medicii să identifice ultimele statistici despre anumite afecțiuni.

Acelaşi agent software permite şi efectuarea plății serviciilor medicale, oferite de un anumit furnizor. Fiabilitatea acestor tipuri de aplicații este de importanță majoră. Fiabilitatea fiecărei componente a sistemului, care include şi *m*-aplicația, este foarte importantă pentru fiabilitatea întregii aplicații.

2. Tranzacții utilizate in dezvoltarea de *m*-aplicații

În [KERS02] se prezintă o comparație între şaptesprezece sisteme ce oferă tehnologii suport pentru decizii folosite în *e*-negotiation. Majoritatea sistemelor tind să se concentreze asupra tranzacțiilor, folosind software structurat pe un set de procese. Doar şase din acestea au mecanism suport pentru schimburi de informații între părțile implicate, incluzând e-mail sau funcții de chat.

În continuare, sunt evidențiate trei funcții pe care sistemele suport pentru tranzacții şi negocieri trebuie să le aibă:

- *procesul de achiziționare a cunoştințelor din informații* vizează termenele limită şi relevanța informației determinante pentru calitatea deciziilor economice; *m*-aplicațiile utilizate în negocieri, licitații trebuie să fie ghidate spre căutarea, procesare şi livrarea de informații senzitive la context, care să ofere avantaje competitive utilizatorului;
- *procesul de comunicare* se referă la abilitatea de a folosi rețelele digitale, pentru a furniza comunicația în ambele sensuri a părților implicate în *m*-commerce;
- *procesul tranzacțional* permite utilizatorilor să execute, în ordinea oportună, operații sau activități pentru a finaliza anumite negocieri; exemple de astfel de activități constau în căutarea şi filtrarea informațiilor,

calculul soldului dintr-un cont bancar şi transfer între conturi bancare, generarea de contracte de afaceri, verificarea şi autentificarea părţilor implicate într-o negociere sau în *m*-commerce etc.

O platformă robustă şi fiabilă compusă din *m*-computing şi *m*-application oferă toate aceste trei categorii de funcţii într-un mod coordonat, sincronizat, fiabil şi oportun. Accesul la resurse distribuite, care se realizează printr-o tranzacţie, se numeşte tranzacţie distribuită. În aceste sens, un exemplu îl constituie scrierea în două baze de date diferite, a informaţiilor privitoare la soldul bancar.

M-aplicaţiile, în sensul celor care rulează pe un dispozitiv mobil, interacţionează cu alte sisteme distribuite pentru a rezolva anumite sarcini prin tranzacţii. O tranzacţie este definită sub forma unei colecţii de operaţii asupra unei stări a aplicaţiei. Colecţia de operaţii are proprietăţile de atomicitate, consistenţă, izolare şi persistenţă, fiind referite pe scurt sub denumirea de ACID.

3. Caracteristici ale tehnologiilor wireless pentru *m*-aplicaţii

Dispozitivele mobile, pentru care se construiesc *m*-aplicaţii sunt: telefoane mobile, PDA-urile (Personal Digital Assistant), laptop - urile, GPS etc. Infrastructura în care se desfăşoară o *m*-aplicaţie este formată, în afară de dispozitivele mobile, din: protocoale, standarde şi limbaje de programare.

Standardele curente şi viitoare pentru reţelele de telefonie mobilă sunt prezentate în tabelul de mai jos.

Tabelul nr. 1 Standarde pentru reţelele de telefonie mobilă

Denumire standard	Descriere	Comentarii
GSM (Global System for Mobile communications)	A doua generaţie de reţea de comunicaţie pentru telefoane digitale celulare s-a stabilit pe întreg mapamondul, exceptând Nordul Americii şi Japonia. Viteza este între 9.6 şi 14.4Kbps. (kilo biţi pe secundă). Pentru controlul accesului la resursa canal de comunicaţie foloseşte metoda TDMA)	Pentru că GSM este aproape universal (şi costuri ridicate de trecere la altă tehnologie), europenii nu vor să-l înlocuiască cu un standard mai nou şi mai rapid. Rezultatul este îmbunătăţirea adusă la GSM prin (GPRS, UMTS) care sunt din ce în ce mai populare deoarece nu aduc modificări majore la sistemul de bază.
GPRS (General Packet Radio Services)	Îmbunătăţire la GSM. Proiectat pentru a fi folosit ca o treaptă între telefoanele de generaţie a doua -2G şi a treia -3G (2.5G).	Multe companii aşteaptă trecerea directă la 3G pentru a evita costurile de îmbunătăţire. Vitezele de transfer sunt de până la 144kbps.
PDC (Personal Digital Cellular)	Standard de telecomunicaţie mobilă folosit în Japonia.	Foloseşte tehnologia TDMA şi se bazează pe American DAMPS.
TDMA (Time Division Multiple Access) *CDMA* (Code Division Multiple Access)	Sunt tehnologiile folosite în SUA. Reţelele de comunicaţii folosite în SUA folosesc pentru controlul accesului la resursa canal de comunicaţie CDMA şi mai puţin TDMA. TDMA este folosit de GSM	Pista de lansare pentru tehnologiile WCDMA, UMTS şi CDMA2000 folosite pentru telefoanele de generaţia a treia 3G.
UMTS (Universal Mobile Telecommunications System) *W-CDMA* (Wide-band Code Division Multiple Access) *CDMA2000* *EDGE* (Enhanced Data rates for GSM (or Global Evolution)	Standarde pentru telefoanele mobile de generaţia a treia (3G). Generaţia a treia se referă la vârful tehnologic, care poate fi atins acum de telefoanele celulare.	Vitezele vor fi de ordinul sutelor de kbs. Tehnologiile se află încă în cercetare-dezvoltare.

O nouă tendință a producătorilor din domeniul comunicațiilor mobile este de a produce și vinde dispozitive care combină comunicațiile de voce cu cele de date și care permit rularea diverselor aplicații.

Multe firme producătoare de PDA (Palm, Handspring), precum și cele de telefoane mobile (Nokia, Ericsson) au construit dispozitive cu toate facilitățile. Momentan, prețurile sunt încă mari pentru astfel de dispozitive și descurajează m-utilizatorii.

Avantajele oferite de aceste dispozitive sunt:

- mobilitate – acces din orice locație;
- wireless – absența firului care limitează deplasarea;
- disponibilitate – acces în orice moment la informație;
- schimb de date;
- îmbunătățirea comunicațiilor;
- ușor de transportat;
- acces la informația potrivită la timpul potrivit;
- eliminarea utilizării suportului fizic;
- câștig de timp;
- acces personalizat la informație.

Dezavantajele vizează următoarele aspecte:

- interfață limitată pentru operațiile de intrare/ieșire, efectuate cu tastatura/ecranul;
- viteza redusă de transfer a datelor, comparativ cu calculatoarele;
- durata de autonomie a acumulatorului;
- nu există un standard de programare pentru toate telefoanele;
- costul destul de ridicat;
- memoria cu capacitate de stocare limitată și capacitate de procesare redusă, comparativ cu calculatoarele;
- pericolul radiațiilor dispozitivelor pentru care se dezvoltă *m*-aplicații, cu observația că acesta face obiectul unor controverse.

M-aplicațiile vor fi influențate în mod direct de cercetarea și descoperirea unor noi tehnologii wireless, care să permită dezvoltarea *m*-aplicațiilor multimedia și *m*-aplicațiilor bazate pe comunicații.

4. Fiabilitatea în informatică

În proiectarea sistemelor complexe, o funcționalitate ridicată este delegată părții software. Fiabilitatea software a evoluat, ca teorie, din fiabilitatea hardware și este unul dintre parametrii de calitate cei mai importanți.

Fiabilitatea unui sistem se definește ca fiind „măsura succesului cu care un sistem se conformează specificațiilor unei autorități". Similar, reversul fiabilității, *insuccesul*, este apare atunci când „comportarea sistemului deviază de la modul de acțiune, specificat pentru sistem".

Conceptul de insucces privește comportarea sistemului din exterior. Insuccesul vizibil către exteriorul sistemului este denumit eroare. Deci, este corect când se spune că un sistem este tolerant la insuccese, și nu la erori. Cauzele care stau la baza erorilor sunt denumite *greșeli*. Greșelile există într-un sistem, fără a avea loc erori sau insuccese, pentru că circumstanțele în care acest greșeli provoacă erori sau insuccese nu sunt îndeplinite.

Există două activități executate în timpul analizei fiabilității software: *estimare* și *predicție*. În fiecare din cele două activități, sunt folosite în mod imbricat tehnici din statistică, cu modele de fiabilitate, care sunt aplicate insuccesselor din perioada de testare a sistemului.

În principiu, *estimarea* este o retrospectivă, fiind efectuată pentru a determina fiabilitatea pe o perioadă de timp din trecut. Activitatea de *predicție* parametrizează modelele de fiabilitate, folosite pentru estimare, și utilizează datele disponibile pentru a preconiza fiabilitatea într-o perioadă de timp viitoare.

În general, modelele de fiabilitate software se clasifică ca fiind modele *black box* (cutie neagră) și modele *white box* (cutie albă). Diferența între cele două este simplă, primele ținând cont de structura aplicației, în timp ce modelele black box nu țin cont de acest aspect.

Modelele de fiabilitate software black box reprezintă o categorie specială, adaptată noilor condiții.

În procesul de dezvoltare software, este tipic ca produsul final să aibă defecte de proiectare. Pentru un anumit set de intrări, aceste defecte sunt activate, rezultând abateri ale produsului software de la specificațiile formulate, denumite insuccese.

O dată un insucces detectat, prin procesul de testare și pus în corespondență cu greșeala care-l generează, este remediat prin repararea greșelii. Procesul de reparare a greșelilor, dacă nu introduce noi greșeli, mărește fiabilitatea produsului. Dacă datele privind insuccesele sunt înregistrate sub formă de număr de insuccese într-o perioadă de timp sau ca interval de timp între insuccese, exprimat în secunde, se folosesc modele statistice pentru a identifica trendul fiabilității.

Astfel de modele sunt cunoscute pentru creșterea fiabilității software (SRGMs– software reliability growth models), fiind folosite în activitățile de estimare și predicție a fiabilității software. Modelele black box nu iau în considerare structura internă a aplicației în estimarea și predicția fiabilității, aplicația fiind considerată o entitate monolitică.

În secțiunile următoare, este prezentat modelul pentru creștere fiabilității software Jelinski-Moranda de-eutrophication.

Unii dintre cei mai utilizați termeni în descrierea modelelor de fiabilitate software sunt prezentați în tabelul următor.

Tabelul nr. 2 Termeni utilizați în descrierea de modele de fiabilitate software

Termen	Explicație
$M(t)$	Numărul total de insuccese înregistrate într-o perioadă de timp t.
$\mu(t)$	Funcția valoare medie pentru un SRGM. Aceasta reprezintă numărul de insuccese așteptate $\mu(t) = E[M(t)]$.
$\lambda(t)$	Intensitatea insucceselor, calculându-se ca derivata funcției valoare medie. Deci, $\lambda(t) = \mu'(t)$.
$Z(\Delta t/t_{i-1})$	Rata întâmplării, care reprezintă densitatea de probabilitate când se experimentează al i^{lea} insucces înregistrat la $t_{i-1}+\Delta t$, față de al $(i-1)^{lea}$ insucces care a avut loc la t_{i-1}.
$z(t)$	Rata de hazard per greșeală, care reprezintă probabilitatea ca o greșeală să nu fie activată și ar fi cauzat un insucces dacă ar fi fost activată. Acest termen, în principiu, este considerat constant (φ) de mai multe modele.
N	Numărul inițial de greșeli, prezent în aplicație înainte de a fi testată.

În general, datele de intrare, furnizate SRGMs-urilor, sunt fie perioade între insuccese $\{\Delta t_1, \Delta t_2, \Delta t_3, \ldots\}$, fie momentul în care insuccesul a avut loc $\{t_1, t_2, t_3, \ldots\}$. Toate modelele prezentate în secțiunile următoare presupun independența între insuccese. Există cadre [15] care încorporează dependențe statistice între insuccese.

Modelul Jelinski-Moranda este extins pentru noua tipologie de aplicații mobile.

Dezvoltat în 1972, modelul Jelinski - Mornada este unul din primele modele de fiabilitate software.

Ipotezele modelului sunt:

- N greșeli inițiale în cod sursă înainte de testare sunt fixate și cunoscute;
- insuccesele nu sunt corelate, iar timpii dintre insuccese sunt independenți și distribuiți aleator exponențial;
- îndreptarea greșelilor este instantanee, și nu introduce nici o greșeală în aplicația ce este supusă testului;
- rata de hazard $z(t)$ al fiecărei greșeli este invariabilă în timp și este constantă (φ). Mai mult, fiecare greșeală este egală din punct de vedere al generării insucceselor.

Ipotezele conduc la calculul ratei hazardului $Z(\Delta t/t_{i-1})$, după îndepărtarea a celei de a $(i-1)$ greșeală, ca fiind proporțională cu numărul de greșeli rămase în aplicație $(N - M(t_{i-1}))$.

$$Z(\Delta t/t_{i-1}) = \varphi*(N - M(t_{i-1})) \quad (1)$$

Funcția valoare medie și funcția intensitatea insucceselor este:

$$\mu(t) = N*(1-e^{-\varphi t}) \quad (2)$$

$$\lambda(t) = N*\varphi*e^{-\varphi t} = \varphi*(N - \mu(t)) \quad (3)$$

Fiabilitatea software, obținută prin acest model, este expresia:

$$R(t_i) = e^{-\varphi(N-(i-1)ti)} \quad (4)$$

Modelul cere timpul trecut între insuccese sau timpi actuali de insuccese pentru a estima parametrii modelului.

Modelele de fiabilitate software *white box* iau în considerare structura internă a aplicației în estimarea fiabilității. Modelele *black box* iau în considerare doar interacțiunile întregii aplicații cu mediul în care se desfășoară, fiindcă sunt inadecvate a fi întrebuințate la aplicațiile și sistemele software, bazate pe componente.

Susținătorii modelelor white box afirmă că modelele care iau în considerare fiabilitățile componentelor dau estimări mult mai reale asupra fiabilității întregului sistem.

În modelele white box, componentele și modulele sunt identificate, proiectate și testate independent. Insuccesul sistemului se estimează în funcție de arhitectura sistemului și este în funcție de fiabilitățile fiecărei component în parte. Modul în care comportamentul cauzat de greșeli este combinat cu arhitectura sistemului generează trei clase de modele de fiabilitate de tip white-box:

- modele bazate pe cale;
- modele bazate pe stare;
- modele aditive.

În continuare, este descris modelul Krishnamurthy și Mathur, care face parte din prima categorie de modele de tip white box.

Prima ipoteză a acestui model [12] este că fiabilitățile componentelor sunt cunoscute. O cale este drumul pe care un set de intrări îl „parcurge" prin program. Fiabilitatea căilor este calculată din fiabilitatea secvenței de componente utilizate, pe care un test dintr-un set de teste se rulează.

Arhitectura aplicației este esențială și este determinată de secvența de componente utilizate de testele sau simulările executate.

Dacă fiecare componentă are o fiabilitate R_i, în ipoteza că fiecare componentă „cade" independent, atunci fiabilitatea căii urmate, R_p, a unei urme $M(P, t)$ pentru un program P, care conține o secvență de m componente executate în urma unui test t dintr-un set de teste T, este:

$$R_p = \prod_{m \in M(P,t)} R_i \quad (10)$$

Fiabilitatea întregului sistem R_{sys} este media aritmetică a fiabilității fiecărei căi în parte:

$$R_{sys} = \sum_{j=1}^{n} Rp_j \Big/ |T| \quad (11)$$

unde:

- *n este numărul de căi;*
- *Rp_j este fiabilitatea fiecărei căi în parte.*

Dependența între componente introduce posibilitatea ca multiple evenimente ce au loc pe parcursul unei căi să nu fie independente.

Alte modele bazate pe căi au abordări similare, iar fiabilitatea sistemului este calculată, în general, urmând toate căile posibile din sistem prin testare/experimentare sau printr-un program special construit.

5. Fiabilitatea în aplicația MiniStealth

Pentru a ilustra problemele de fiabilitate, se construiește un sistem distribuit, care are clienți *m*-aplicații. *M*-aplicația client se conectează la un server și execută operațiile de autorizare și autentificare a accesului, pentru a obține accesul la un serviciu oferit de sistemul distribuit (serviciu medical, bancar sau financiar etc).

M-aplicația, în sensul strict de aplicație care rulează pe un telefon mobil, mai este denumită și *midlet* în această lucrare, fiind dezvoltată peste MIDP 1.0. Din midlet se deschide o conexiune folosind protocolul HTTP la un server. Midlet-ul trimite numele utilizatorului, parola, un număr aleator și o amprentă de timp. Din aceste

informații, se construiește un hashcode, printr-o funcție de dispersie cunoscută, de exemplu SHA1. Serverul primește numele, amprenta, numărul aleator și hashcode-ul. Într-o bază de date există, de asemenea, criptată parola utilizatorului.

Se extrage parola din baza de date, se decriptează și se construiește un nou hash din parola decriptată și informațiile primite de la midlet. Dacă hashcode-ul construit este identic cu cel primit, utilizatorul va folosi serviciile oferite de server, dacă nu este respins.

Acest sistem distribuit nu este fiabil. De exemplu, dacă mașina pe care se află serverul de aplicație nu mai funcționează, utilizatorul nu mai achiziționează serviciile de care are nevoie.

În continuare, sunt făcute modificări pentru a crește fiabilitatea sistemului în ansamblu și, implicit, pentru a crește fiabilitatea *m*-aplicației. Toate modificările sunt efectuate pentru partea operației de login.

Completarea codului sursă cu preluarea parolei criptate dintr-o bază de date

Servletul, programul care tratează cererile clienților și rulează în containerul serverului de web, trebuie să valideze amprenta de timp, primită de la client. De fiecare dată când clientul încearcă să facă login, servletul trebuie să se asigure că amprenta de timp primită este mai mare decât cea primită anterior.

O altă schimbare este aceea de a înregistra parola într-o locație de memorie în telefon și/sau de a genera cheia secretă din parolă cu ajutorul unei funcții hash și de a folosi un algoritm criptografic simetric (Rijndael) pentru a cripta informațiile cu cheia generată.

La prima vedere, înregistrarea parolei în memoria telefonului nu pare o idee foarte bună. Dispozitivele mici sunt în posesia utilizatorilor lor, iar când aceștia nu mai sunt lângă, acestea sunt blocate. Pierderea sau furtul telefonului sunt similare cu pierderea cardului bancar. Cel care are telefonul trebuie să știe codul și parola de acces la serviciile oferite de server.

Autentificarea este făcută pentru fiecare cerere efectuată de midlet. Modificarea s-a efectuat astfel încât autentificarea să se realizeze doar la intrarea în sistem și la achiziția unor servicii oferite de serverul de aplicații care cer un grad ridicat de securitate. Toate aceste modificări sporesc gradul de încredere în sistemul informatic și, implicit, gradul de încredere în *m*-aplicație. Deci, are loc o creștere a fiabilității sistemului și a *m*-aplicației.

Servletul este rulat de mai multe servere de web în același timp, pe mașini diferite

Pentru a rezolva cererile venite de la midlet către serverul de web, există două variante:

- folosirea unui program software, pentru distribuirea cererilor: se folosește OpenMosix, pentru serverele web ce rulează sub sistemele de operare Unix, Linux: Apache, Tomcat, Borland Entreprise Server, sau Network Load Balancening, pentru serverele web ce rulează sub Windows: IIS, Apache;
- scrierea în DNS (Domain Name Server) pentru același nume a IP - urilor mașinilor pe care rulează serverul de web; dacă unele din serverele de web sau mașini „cad", sarcinile sunt preluate de calculatoarele ce au IP - ul trecut în DNS: în acest mod, sistemul devine tolerant la insuccese și, deci, crește gradul de fiabilitate al sistemului și al *m*-aplicației.

Servletul pentru procesarea cererilor și oferirea serviciilor

Este folosită procesarea distribuită prin: RMI (Remote Method Invocation), CORBA (Common Object Request Broker Architecture) și tehnologia Jini.

De exemplu, serviciul de cotații bursiere este oferit de mașina 1, 3 și 7. Servletul, pentru a rezolva cererea unui client care vrea serviciul respectiv, lansează o metodă la distanță pe mașina 1. Dacă aceasta nu este disponibilă sau serviciul nu funcționează, se apelează imediat altă metodă la distanță, pentru același serviciu, dar pe mașina 3, și așa mai departe. Aceasta înseamnă o mai mare redundanță de procese și date în sistem, dar sistemul per ansamblu este mult mai fiabil.

Instrumentele criptografice

Întregul sistem folosește instrumente criptografice foarte puternice, iar bazele de date pentru suportul informațional sunt robuste și suportă efectuarea tranzacțiilor.

Fiabilitatea care se referă strict la aplicație este greu de calculat și se face doar pe cale experimentală. Prin programare se testează fiabilitatea pe emulatoare.

Codul sursă ca și explicațiile detaliate despre implementarea fiabilă a acestui sistem distribuit, inclusiv *m*-aplicațiile client, se obțin de la autorii acestui material. Servletul a fost dezvoltat cu J2SDK SE 1.4.1. El rulează în containerul serverului de web Jakarta Tomcat-Apache 4.0.1 Web Server.

Serviciile sunt implementate folosind tehnologiile CORBA și RMI și rulează pe diferite calculatoare într-o rețea TCP/IP. Serviciile financiar și bancar au fost dezvoltate cu J2SDK EE. *M*-aplicația a fost dezvoltată cu Nokia Developer Suite SDK și a fost testată pe un telefon Nokia 6310i, folosind o rețea adecvată. Pentru a crește fiabilitatea și interoperabilitatea serviciilor WAP pentru o companie de telefonie mobilă, se dezvoltă servicii folosind tehnologia CORBA.

6. Concluzii

Construirea de software fiabil nu este o activitate ușoară. Această problemă trebuie atacată la câteva niveluri: cerințe utilizator, analiză, proiectare și implementare. În anumite situații, clienții nu au nevoie de aplicații foarte complexe. Deseori, inginerii software găsesc acele specificații critice și negociază pe ele, scăpând de cerințele neimportante, care cresc complexitatea proiectului și cer multe resurse. Scopul principal este reducerea complexității. Cerințele „ascunse" trebuie expuse și înțelese. De multe ori, problema esențială a sistemului rămâne nerezolvată, făcând sistemul inacceptabil.

Există mai multe arhitecturi complexe pentru care trebuie analizată fiabilitatea sistemului: arhitectura logică, arhitectura fizică, arhitectura proceselor, arhitectura dezvoltării.

Procesul de simplificare a programelor la nivelul proiectare are menirea de a găsi metodele redundante, de a valorifica componentele reutilizabile, de alege cei mai buni algoritmi pentru aflarea soluțiilor problemei de rezolvat, stabilirea șabloanelor standard pentru interfețe și procese, identificarea fluxului de date dinamic etc.

Principalul obiectiv la acest nivel este stabilitatea sistemului. Stabilitatea proceselor alocate pentru rezolvarea problemelor trebuie studiată. Adesea, depășirile de memorie și de resurse alocate subminează stabilitatea și fiabilitatea sistemului. Stabilitatea algoritmului trebuie validată tot în această etapă.

La nivelul implementării, standardele de codificare și programare furnizează metode și instrumente pentru a face sistemele să ruleze fiabil și repede.

Prin metode specifice analizei, proiectării și implementării se urmărește creșterea calităților produsului software, sistemului, a *m*-aplicației, în particular. Una din calitățile importante este și va fi fiabilitatea.

Bibliografie

1. **BISHOP, P.G., R. BLOOMFIELD**: A Conservative Theory for Long Term Reliability Growth Prediction. În: IEEE Transactions on Reliability, Vol. 45, No.4, December 1996, pp. 550-560.
2. **BISHOP, P.G., R. BLOOMFIELD**: Worst Case Reliability Prediction on a Prior Estimate of Residual Defects. În: Proc. of the 13th IEEE International Symposium on Software Reliability Engineering (ISSRE-2002), November 2002 pp. 295 – 303.
3. **BUI, T., J. ONDRUS**: *M*-computing for Real-time Negociation Support. Workshop September 2002, USA.
4. **GOETZ, B**.: Java Theory and Practice: Understanding JTS – An Introduction to Transactions, IBM, USA 2002.
5. **GOKHALE, S., T. PHILIP, P. MARINOS, K. TRIVEDI**: Unification of Finite-failure Non-homogenous Poisson Process Models Through Test Coverage. În: Proc. of the 7th IEEE International Symposium on Software Reliability Engineering (ISSRE-96), November 1996.
6. **GOKHALE, S., W. E. HONG, K. TRIVEDI, J.R. HORGAN**,: An Analytical Approach to Architecture based Software Reliability Prediction. În: Proc. of the 3rd IEEE International Computer Performance and Dependability Symposium, (IPDS-98), 1998, pp. 13-22.
7. **GROTTKE, M**.: Software Reliability Model Study", PETS Project Technical ReportA.2, University of Erlangen-Nuremberg, 2001.
8. **IONESCU, T., I. HOHAN, A. HOHAN**: Software Reliability and its Reference to Certification Criteria. Software reliability workshop 2001, Inforec Publishing House, Bucharest, 2001.

9. **IVAN, I., P. POCATILU, C. TOMA:** A. Leau: e3-com. În: Informatică Economică, Nr. 3(19)/2001, Bucureşti, 2001.

10. **IVAN, I., L. TEODORESCU**: Managementul calităţii software, Editura Inforec, Bucureşti, 2001.

11. **KERSTEN, G.**: The Science and Engineering of *E*-negotiation: Review of the Emerging Field, *INR05/02*, University of Concordia, Working Paper, 2002.

12. **KRISHNAMURTHY, S., A. P. MATHUR**: On the Estimation of Reliability of a Software System Using Reliability of its Components. În: Proc. of the 8th IEEE International Symposium on Software Reliability Engineering (ISSRE'97), November 1997, pp. 146 – 155.

13. **LYU, M. R.** (Ed.): Handbook of Software Reliability Engineering. În: IEEE Computer, Society Press, 1996.

14. **PHAM, H.**: Software Reliability, Springer-Verlag, 2000.

15. **POPSTAJANOVA, K., K. TRIVEDI**: Failure Correlation in Software Reliability Models. În: IEEE Transactions on Reliability, Vol. 49, No. 1, March 2000.

16. **POPSTAJANOVA, K., K. TRIVEDI**: Architecture Based Approach to Reliability Assessment of Software Systems, Performance Evaluation, Vol. 45, No.2, June 2001.

17. **VOGLER, H., M. L. MOSCHGATH, T. KUNKELMANN, J. GRUNEWALD**: The Transaction Internet Protocol in practice: Reliability for WWW Applications", Darmstadt, Deutschland 2000.