Verificarea şi specificarea formală a produselor software. Posibilități de implementare şi dificultăți.

Mariana Catruc, Radu Melnic, Ion Balmuş
Universitatea Tehnică a Moldovei
mariana.catruc@ati.utm.md, rmelnic@mail.utm.md, balmus@mail.utm.md

***Abstract* — Formal specification is concerned with producing an unambiguous set of product specifications so that customer requirements, as well as environmental constraints and design intentions, are correctly reflected, thus reducing the chances of accidental fault injections. The purpose of this article is to describe the formal analyses and verifications based on formal models of program and their expected behavior, as an alternative way for software quality assurance.**

***Cuvinte cheie* — V&V, Specificație formală, Testare Statică, Asigurarea Calității, FSM**

I. INTRODUCERE

Verificarea formală constă în verificarea corectitudinii sau absenței erorilor unui program sau unei arhitecturi conform specificațiilor formale. Astfel, existența specificațiilor formale este o condiție obligatorie pentru verificările formale. Atât tehnicile de specificare formală cât şi tehnicile de verificare formală se referă la metodele formale de verificare. În timp ce metodele formale sunt folosite pentru dezvoltarea produselor software, specificațiile formale sunt folosite pentru analiza cerințelor şi a specificațiilor software. Verificările şi analizele formale sunt folosite pentru verificarea arhitecturii şi codului înaintea activităților de verificare şi validare (V&V) adiționale [6].

Folosirea metodelor formale poate fi prezentată ca un proces format din două etape:

1. *Elaborarea specificațiilor formale:* Comportamentul aşteptat şi trăsăturile produselor software sunt prezentate sub forma de modele formale. Modelele formale ale programului, arhitecturii şi comportamentului sunt specifice produsului software, şi pot fi adecvat construite pentru verificări formale ulterioare. Pentru a reduce costurile şi a mări beneficiile din dezvoltarea metodelor formale, aceste modele pot fi aceleaşi ca şi specificațiile formale sau adaptate prin formalizarea specificațiilor informale ale produsului.

2. *Efectuarea verificări formale:* Tehnicile de analiză formală sunt aplicate pe componentele produsului pentru a verifica corectitudinea lor conform specificațiilor formale proprii, sau pentru a verifica anumite proprietăți. Aceste tehnici sunt de obicei organizate ca framework-uri independente de produs cu reguli care servesc pentru analizele formale. Cel mai cunoscut tip de reguli este o mulțime de *axiome* folosite în verificarea corectitudinii [1].

II. SPECIFICAȚIILE FORMALE

Specificația formală are ca scop producerea unei mulțimi riguroase de specificații ale produsului astfel ca cerințele beneficiarului precum şi restricțiile față de mediu şi conceptele arhitecturale sunt corect reflectate, astfel pot fi reduse şansele apariției unor defecte accidentale. Specificațiile formale se axează pe aspectul funcțional sau pe corectitudinea comportamentului programului şi nu pe aspectele nefuncționale. Cu specificațiile formale, caracteristicile specificațiilor cerințelor software (completitudine, claritate, consistență) pot fi mai uşor analizate formal utilizând diverse tehnici de verificare. Specificațiile formale pot fi produse în diferite forme, şi pot fi împărțite în două categorii generale, specificații descriptive şi specificații operaționale, după cum urmează:

- *Specificațiile descriptive* se concentrează asupra proprietăților şi condițiilor asociate cu produsul software şi componentele lui. De exemplu:
  - Diagramele entitate-relație sunt des folosite pentru descrierea componentelor şi interfețelor produsului.
  - Specificațiile logice se axează pe proprietățile formale asociate cu diferite componente ale produsului sau cu produs propriu zis.
  - Specificațiile algebrice se axează pe calculele funcționale realizate de un program sau un segment de program.
- *Specificațiile operaționale* se concentrează asupra comportamentul dinamic a sistemelor software. De exemplu:
  - Diagramele fluxului de date, care specifică fluxul informației între unitățile funcționale principale.
  - Diagramele UML - specifică comportamentul individual al obiectelor sau al componentelor produsului.

– Maşinile cu stare finită (FSM) - descriu fluxul de control în tranziţiile de stare.

Majoritatea specificaţiilor operaţionale pot fi testate folosind tehnici de testare clasice şi analizate cu diferite tehnici de analiză statică.

III. VERIFICAREA ŞI ANALIZA FORMALĂ

Verificările formale verifică conformitatea arhitecturii sau codului software cu specificaţiile formale. Pentru realizarea verificărilor formale sunt necesare un set de metode şi reguli pentru argumentare. Cele mai cunoscute şi răspândite includ: corectitudinea axiomatică [1], cele mai slabe precondiţii (weakest pre-conditions) [2] şi corectitudinea funcţională [5]. Ideile de bază sunt următoarele:

- Aproximarea *axiomatică* se referă la specificaţiile logice a programelor sau la arhitectura formală prin asocierea fiecărui tip de program sau element de design cu o axiomă prin transformările logice a stării programului înainte şi după execuţia acestuia. Această aproximare poate produce o dovadă de corectitudine a unui program, segment de program sau design formal, ţinându-se cont de specificaţiile formale proprii.
- Abordarea *„cea mai slabă pre-condiţie”* se axează pe scop sau pe rezultatul calculat care este capturat de starea finală a secvenţei de execuţie. O serie de operaţii înlănţuite transformă starea finală şi proprietăţile ei într-o stare iniţială şi proprietăţile asociate, care verifică corectitudinea obiectului de verificare dacă proprietăţile stării iniţiale pot fi satisfăcute.
- *Corectitudinea funcţională* sau *calcule de program (program calculus)* este similară aproximării axiomatice în sensul că axiomele elementelor de program sunt predefinite. Execuţiile simbolice sunt folosite pentru a conecta aceste elemente într-un program. Corectitudinea programului, sau funcţia matematică specificată şi calculată de program, este verificată prin acest proces.

Fiecare dintre aproximările de mai sus produc o demonstraţie a corectitudinii totale a programului. Uneori poate fi acceptată o demonstraţie parţială sau verificarea anumitor proprietăţi în loc de o demonstraţie totală pentru a reduce costurile de verificare însă acoperind trăsăturile sau proprietăţile importante ale produsului sau al mediului aplicaţiei.

IV. APLICABILITATE ŞI IMPLEMENTARE

Cel mai mare neajuns în metodele formale este costul mare asociat cu sarcina dificilă de executare a activităţilor intensive umane în mod corect fără un suport automatizat adecvat. Chiar dacă verificarea de model şi unele tehnici de analiză formală au ameliorat problema la unele nivele, metodele formale în general încă rămân a fi o alternativă costisitoare pentru asigurarea calităţii produselor software. Acesta şi alţi factori afectează aplicabilitatea şi eficacitatea tehnicilor de verificare formală în procesul de inginerie calitativă.

Problema aplicabilităţii poate fi descrisă luând în consideraţie următoarele:

1. Tipul de produse sau domenii de aplicare, care ar beneficia din folosirea tehnicilor de verificare şi analiză formală.
2. Tipul de elemente software din produsele de mai sus care pot fi verificate sau analizate cu aceste tehnici.

Orice produs poate beneficia din folosirea acestor tehnici formale, însă din cauza experienţei cerute de la personalul implicat în activităţile de verificare şi analiză formală cât şi din cauza costului mare, aceste tehnici sunt cel mai des folosite în programe mici, sau într-o submulţime mică de elemente ale unui sistem software mai mare care cere calitate înaltă sau unde pierderile în caz de eşec ar fi substanţiale. Aşa sisteme software mari includ sistemele de siguranţă înaltă, sau componente şi funcţii critice pentru sistemele software mari. Chiar şi pentru sisteme de siguranţă înaltă, tehnicile de verificare formală şi analiză sunt de obicei folosite selectiv şi nu uniform pe toate componentele sistemului. De obicei activităţile de verificare sunt realizate după faza de dezvoltare şi se bazează pe rezultatele obţinute din analizele aleatoare.

Cele mai des analizate sau verificate obiecte sunt codurile programelor. Totuşi, orice document tehnic poate fi verificat sau analizat formal, atât timp cât pot fi construite specificaţii formale pentru el.

Tehnici similare pot fi de asemenea folosite cu succes în verificare şi analiza programelor distribuite, sistemelor hardware şi funcţiilor lor, reţelelor de comunicare şi protocoalelor, etc. Ideea de bază este formalismul distribuit şi limbajul formal care pot fi folosite pentru a descrie, modela şi analiza diferite tipuri de sisteme, nu doar sisteme software.

V. INTEGRARE ŞI LIMITĂRI

Tehnicile verificării formale sunt dependente de specificaţiile formale ale sistemului (afirmaţii logice şi algebrice, definiţii operaţionale, sau folosirea automatelor cu stare finită (FSM)). Integrarea acestor tehnici de specificare, verificare şi analiză produce diferite metode formale, aşa ca dezvoltarea pas cu pas şi verificarea bazată pe specificaţii logice şi demonstrarea axiomatică a corectitudinii. În majoritatea acestor metode formale, dezvoltarea şi verificarea software funcţionează împreună pentru dezvoltarea produselor şi componentelor software de calitate înaltă.

Există anumite limitări în sisteme formale, în primul rând legate de simplificarea realităţii fizice în procesul abstractizării şi apoi dificultăţi legate de limbaj şi limitările hardware. Prin urmare, metodele formale nu pot garanta perfecţiune pentru software, dar pot însă asigura corectitudinea în raport cu specificaţiile formale şi proprietăţile verificate. Aceasta depinde de construcţia fără erori a demonstraţiilor de corectitudine sau de performanţa, analizelor formale, care nu pot fi garantate. În consecinţă,

metodele formale nu pot fi aplicate la toate activitățile de asigurare a calității. În schimb, ele pot fi folosite împreună în dezvoltarea software și în procesele de mentenanță, sau integrate în metodologii și/sau tehnologii software.

Un exemplu concret a acestei integrări este tehnologia descrisă de Mills [3], care include două componente importante:

- Verificarea formală bazată pe corectitudinea funcțională este folosită în timpul proiectării și implementării produsului.
- Testarea statistică bazată pe lanțul operațional de profil a lui Markov [4] este folosită în stadii mai tardive ale dezvoltării produsului.

În general, metodele formale sunt eficiente în dezvoltarea produselor software și a mediilor, unde cerințele pot fi captate sub formă de specificații formale și rămân stabile permițând rafinamente și verificări mai avansate. Din fericire, majoritatea subsistemelor software folosite în sistemele de înaltă siguranță fac parte din această categorie, astfel ele pot beneficia de aceste metode formale. Pentru produse din mediul volatil de piață, schimbările cerute des și îmbunătățirile soluțiilor fac folosirea metodelor formale dificilă.

Verificarea și analiza formală este cel mai des asociată cu inspecțiile formale și analiza statică ca activități de asigurate a calității. Unele variații a tehnicilor de analiză și verificare formală sunt de asemenea asociate cu testarea. De exemplu, execuția simbolică în verificarea corectitudinii este similară cu testarea de modul cu variabile simbolice în locul valorilor reale pentru a produce rezultate generalizate după o singură execuție a testului.

În contextul strategiilor de asigurare a calității metodele formale aparțin categoriei de prevenire a defectelor și sunt aplicate asupra specificațiilor formale eliminând diferite surse de erori din cauza înțelegerii incorecte a cerințelor produsului. Când demonstrația de corectitudine nu poate fi construită aceasta poate fi un indiciu de prezență a defectelor, astfel excluzând activități adiționale pentru detectarea și eliminarea defectelor. Verificarea de model și alte analize formale sunt mai aproape de inspecție și testare, în sensul că pot identifica direct problemele în timpul analizelor. Așa informație poate apoi fi folosită pentru localizarea și fixarea defectelor.

Ca o strategie de prevenire a defectelor, analiza și verificarea formală cere un efort destul de mare, și desigur cunoștințe matematice și logice în efectuarea verificărilor sau analizelor. Demonstrațiile de corectitudine sunt dificil de realizat și sunt voluminoase, chiar și pentru programe scurte. Imposibilitatea de a automatiza total aceste demonstrații fac ca procesul manual pentru orice program de mărime rezonabilă va fie predispus la greșeli.

În consecință, instruirea și antrenarea sunt cheia spre succesul acestor tehnici. Totuși, acest efort nu trebuie limitat doar la oamenii care realizează activitățile de verificare și analiză, dar și la managerii software, clienți și alte părți implicate astfel ca rezultatele să fie interpretate corect, strategiile corespunzătoare să poată fi selectate din timp, resursele necesare să fie alocate, și activitățile paralele de asigurare a calității să poată fi îndeplinite pentru a complementa folosirea acestor tehnici. Unii practicieni și savanți au observat de asemenea că metodele formale cer responsabilități și stres mult mai mare de la dezvoltătorii de software, pe când instruirile și training-urile pot ajuta la reducerea acestui stres și pot face aplicarea metodelor formale mult mai reușite.

VI. CONCLUZII

Tehnicile de verificare și analiză formală, în special când sunt incluse în metode formale și tehnologii corespunzătoare de dezvoltare, au multe avantaje. Metodele formale nu sunt doar efective în asigurarea unei calități mai înalte pentru multe aplicații, dar de asemenea sunt mult mai efective decât strategiile alternativele de asigurare a calității în unele circumstanțe.

Totuși, ca o alternativă a asigurării calității, tehnicile de verificare și analiză formală nu au încă popularitate și o utilizare mai largă în afara inspecției și testării. Pe lângă propriile lor limitări legate de cost, proces și alte probleme, cea mai importantă cauză este lipsa de experiență cerută, care poate fi parțial soluționată și îmbunătățită în cadrul universitar. Un motiv pentru scrierea acestui articol a fost dorința de a promova înțelegerea tehnicilor de verificare formală ca o alternativă viabilă pentru profesioniștii software și motivarea de a le folosi în asigurarea calității și siguranței sistemelor software.

Specialiștii trebuie să ia în considerație mulțimea de avantaje pe care le oferă metodele formale în general, și tehnicile de verificare formală în particular. Metodele formale merită a fi tratate ca un element potențial ce poate fi combinat într-o strategie completă pentru atingerea obiectivelor de calitate în cadrul proiectelor, în particular pentru cei ce doresc un nivel înalt de calitate.

VII. REFERINȚE BIBLIOGRAFICE

[1] Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. М.: БИНОМ, 2008, 368с. ISBN 978-5-94774-825-3.

[2] Tudor Bălănescu. Corectitudinea algoritmilor. Editura Tehnică, București, 1995. ISBN: 973-31-0895-2.

[3] Mills H., Basili V., Gannon J., and Hamlet R. Principles of Computer Programming: A Mathematical Approach. Ala and Bacon, Inc., Boston, 1987.

[4] Catruc M., Mironiuc A. *Testarea statistică web utilizând Modele Markov Unificate (UMM)*. Culegerea lucrărilor Conferinței Internaționale „Telecomunicații, Electronică și Informatică". Chișinău, UTM, mai 17-20, 2012, p.367-371, ISBN 978-9975-45-200-7(vol 1).

[5] Myers G. The Art of Software Testing. *Second edition*. John Wiley & Sons, Inc., Hoboken, New Jersey 2004. ISBN: 0-471-46912-2.

[6] Tian Jeff. Software Quality Engineering. Testing, Quality Assurance, and Quantifiable Improvement. John Wiley & Sons, Inc., Hoboken, New Jersey 2005. ISBN: 0-471-71345-7.