Modeling software viability based on design and minor changes of object oriented code

Lect.dr. Marian CRISTESCU, prep. Daniel HUNYADI, prep. Mircea MUŞAN
Catedra de Informatică Economică, Universitatea „Lucian Blaga" Sibiu

Object-oriented design and development is becoming very popular in today's software development environment. Object oriented development requires not only a different approach to design and implementation, it requires a different approach to software metrics. Since object oriented technology uses objects and not algorithms as its fundamental building blocks, the approach to software metrics for object oriented programs must be different from the standard metrics set. Some metrics, such as lines of code and cyclomatic complexity, have become accepted as standard for traditional functional/ procedural programs, but for object-oriented, there are many proposed object oriented metrics in the literature.
Keywords: *object-oriented technology, software design, software reliability, component reusability, metrics of mentenability, quality estimation and modeling.*

Introducere

Ca urmare a popularității operației de reutilizare a software-ului, componentele sunt reutilizate în trei moduri:

- aşa cum este sau cutie neagră;
- schimbare minoră sau cutie gri;
- schimbare majoră sau cutie albă.

Primele două sunt dezirabile într-un proces de reutilizare desăvârşit, bazat pe dezvoltarea software. Al treilea necesită informații referitoare la caracteristicile interne ale componentei. Reutilizarea tip cutie neagră se referă numai la interfața componentei.

Conform [DIET99], atributele de nivel înalt ale calității cum sunt fiabilitatea sau mentenabilitatea nu se construiesc direct în software; trebuie identificat şi construit un set consistent, armonios şi complet de proprietăți ale sistemului de programe, cum sunt programele şi modulele fără efecte colaterale, care să se transforme în manifestări ale fiabilității şi mentenabilității.

Model de estimare a calității pentru designul orientat pe obiecte

Modelul de calitate pentru designul orientat pe obiecte QMOOD (**Q**uality **M**odel for **O**bject **O**riented **D**esign) este reprezentativ pentru această clasă şi a fost folosit în studiul fiabilității sistemelor de programe. **QMOOD** este utilizat pentru evaluarea atributelor externe de calitate, precum funcția de reutilizare şi flexibilitatea designului orientat pe obiecte şi are la bază proprietățile interne ale componentelor designului [BANS97].

În cadrul acestui model, proprietățile tangibile ale designului, atât cele structurale cât şi cele funcționale sunt folosite pentru a genera metrici ale designului orientat pe obiecte, care evaluează extinderea proprietăților tangibile în cadrul componentelor acestuia.

Proprietățile tangibile de design ale componentelor şi manifestarea lor într-un sistem de programe contribuie la proprietățile designului orientat pe obiecte. Acestea sunt abstractizarea, încapsularea, cuplarea şi coeziunea.

Modelul relaționează proprietățile designului orientat pe obiecte la un set de atribute externe ale calității de nivel înalt. Relațiile sau legăturile de la caracteristicile de design până la atributele externe de calitate sunt valori asignate pe baza importanței contribuției lor la un atribut anume de calitate.

Modelul este validat prin utilizarea unor opinii empirice precum şi ale unor experți, care să fie comparate cu rezultatele obținute prin aplicarea acestuia asupra mai multor sisteme de programe complexe orientate pe obiecte.

Modelul este uşor modificat pentru a include diferite tipuri de metrici ale designului, proprietăți de design, relații de legătură şi atribute de calitate.

În acest fel se asigură un model practic de estimare a calității designului orientat pe obiecte.

Figura 1 prezintă cele 4 stadii şi cele trei mapări - legăturile L_{12}, L_{23} şi L_{34} - folosite pentru conectarea nivelele adiacente în QMOOD.

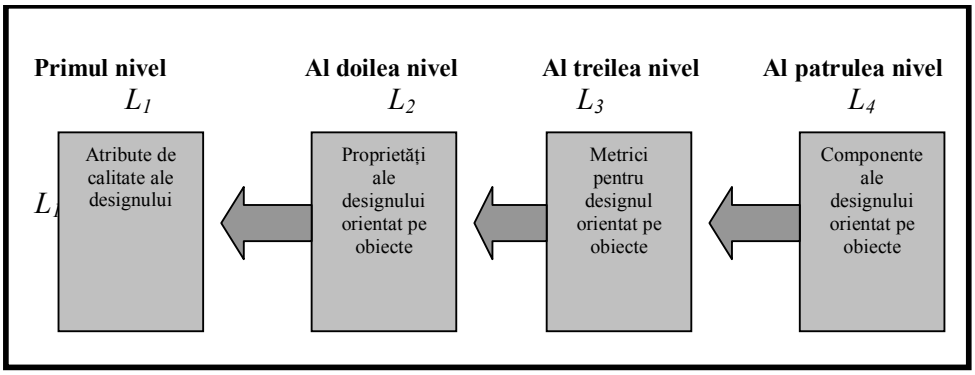

Fig.1. Niveluri de legătură în QMOOD

Dintre proprietățile designului propuse de modelul QMOOD, cele utilizate pentru sistemele de programe sunt: mărimea, abstractizarea, încapsularea, cuplarea, coeziunea, complexitatea, capacitatea de transmitere a mesajelor, compunerea, moştenirea, polimorfismul, ierarhiile de clase.

Fiecare proprietate identificată în model este evaluată prin folosirea uneia sau a mai multor metrici de design. Toate metricile sunt bazate pe informația disponibilă în timpul designului. În acest sens, modelul defineşte un nou set de metrici orientate pe obiecte care sunt bazate numai pe definiția claselor.

În [BANS97], există patru tabele care conțin o listă alfabetică a tuturor metricilor.

Pentru măsurarea proprietăților designului sistemelor de programe au fost selectate metricile prezentate în tabelul următor 1.

Tabelul 1. Metricile asociate proprietăților designului sistemelor de programe

Proprietăți ale designului	Metrici derivate pentru design
Mărimea designului	Mărimea designului în clase (DSC)
Ierarhii	Numărul de ierarhii (NOH)
Abstractizare	Numărul mediu de strămoşi (ANA
Încapsulare	Metrica accesului la date (DAM)
Cuplare	Cuplarea directă a claselor (DCC)
Coeziune	Coeziunea dintre metode în clase (CAM)
Compunere	Măsura agregării (MOA)
Moştenire	Măsura abstractizării funcționale (MFA)
Polimorfism	Numărul de metode polimorfice (NOP)
Capacitatea de transmitere a mesajelor	Mărimea interfeței claselor (CIS)
Complexitate	Numărul de metode (NOM)

Comportamentul extern al unui sistem de programe este influențat de calitatea proprietăților sale interne. QMOOD pune în legătură proprietățile designului şi atributele de calitate prin considerarea relațiilor dintre fiecare proprietate a designului şi atributele de calitate.

Influența fiecărei proprietăți a designului se reflectă asupra atributelor calității şi este pozitivă sau negativă.

Ca rezultat, fiecare atribut de calitate este influențat de mai multe proprietăți ale designului în mod pozitiv sau negativ.

În cazul sistemelor de programe, pentru a păstra simplitatea modelului s-au adoptat două scale de evaluare a semnificației proprietăților pentru determinarea semnificației relative a unei proprietăți a designului asupra

unui atribut de calitate. În cazul influenţelor pozitive a fost utilizată o valoare notată iniţial cu *+1* sau *+0,5*.
Asigurarea rangului selectat pentru valorile calculate ale atributelor de calitate s-a făcut prin ± 1.
Toate valorile iniţiale au fost ajustate proporţional, în aşa fel încât suma noilor valori ale proprietăţilor designului care influenţează atributul de calitate să fie adunată la ± 1.

Tabelul 2 descrie rezultatul semnificaţiei înlănţuirii, bazat pe metoda empirică, în care o valoare influenţată pozitiv prezintă o influenţă pozitivă şi una influenţată negativ prezintă o influenţă negativă.
Valoarea calculată a fiecărui atribut de calitate reprezintă suma ponderilor asociate proprietăţilor designului, potrivit coloanei respective din tabel.

Tabelul 2. Relaţia atribute de calitate – ponderi asociată proprietăţilor designului

	Reutilizabilitate	**Flexibilitate**	**Inteligibilitate**	**Funcţionalitate**	**Extensibilitate**	**Eficienţă**
Mărimea designului	0,50		-0,33	0,22		
Ierarhii				0,22		
Abstractizare			-0,33		0,50	0,20
Încapsulare		0,25	0,33			0,20
Modularitate						
Cuplare	-0,25	-0,25	-0,33		-0,50	
Coeziune	0,25		0,33	0,12		
Compunere		0,50				0,20
Moştenire					0,50	0,20
Polimorfism		0,50	-0,33	0,22	0,50	0,20
Capacitate de transmitere a mesajelor	0,50			0,22		
Complexitate			-0,33			
Suma	1,00	1,00	-1,00	1,00	1,00	1,00

Ecuaţiile pentru calcularea fiecăreia dintre cele 6 atribute de calitate pot fi uşor derivate prin utilizarea datelor din tabelul 2.
Un exemplu de utilizare a informaţiilor din tabelul 2 este calculul valorii atributului reutilizabilitate.

Reutilizabilitate = 0,5 ∗ mărimea designului – 0,25 ∗ cuplarea + 0,25 ∗ coeziunea+ 0,5 ∗ capacitatea de transmitere a mesajelor

De asemenea, toate proprietăţile designului sunt calculate folosind metricile corespunzătoare. Figura 2 prezintă relaţia dintre atributele de calitate, proprietăţile şi metricile designului pentru sistemele de programe.

Concluzii

Reproiectarea interdependenţelor dintre calitate şi proprietăţile designului orientat pe obiecte, pentru sistemele de programe, trebuie să ţină cont de arhitectura acestora şi de profilul operaţional. În practică s-a observat că nivelul de semnificaţie al atributelor de calitate se modifică de-a lungul ciclului de viaţă al sistemelor de programe.

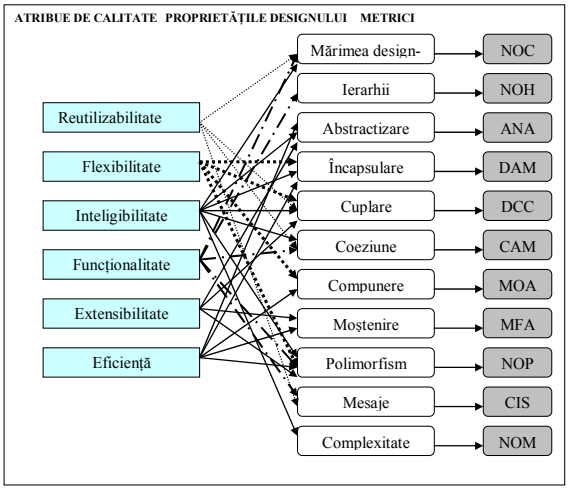

Fig.2. Utilizarea modelului QMOOD pentru sisteme de programe

Modul în care proprietăţile designului sunt relaţionate cu atributele de calitate influen-

țează desfăşurarea procesului de dezvoltare software cu implicații directe asupra calității globale a sistemelor de programe rezultate.

Bibliografie:

[BANS97] - Bansiya J., "*A Hierarchical Model for Quality Assesment of Object- Oriented Design*", Huntsville Press, Alabama, 1997, pag. 57-78;

[DIET99] - Dietrich S.W., Calliss F.W., "*A Conceptual Design for a Code Analysis Knowledge Base*", Software Maintenance: Research and Practice, vol.4, 1999, pag. 19-36;

[HARR98] - Harrold M.J., McGregor J.D., and Fitzpatrich K. J., "*Incremental Testing of Object-oriented Class Structures*", In Proceedings of the 14th International Conference on Software Engineering, pp. 68–80, May 1998;

[KICM00] - Kim S., Clark J.A., and McDermid J. A., "*Class mutation: mutation testing for object-oriented programs*". In Proceedings of the NetObjectDays - Conference on Object-Oriented Software Systems, 2000;

[MAYR00] - Mayrhauserens A., Scheetz M., and Dahlman E., "*Generating test-cases from an object-oriented model with an artificial intelligence planning system*", In IEEE Transactions on Reliability, volume 49, March 2000;

[YACO99] - Yacoub S., Cukic B., Ammar H., "*Scenario-based reliability analysis of component based software*", in: Proceedings of the 10-th International Symposium on Software Reliability Engineering (ISSRE'99), 1999, pag.22-31;