

Arbore de calitate bazat pe strategii de detectie

Ciprian-Bogdan Chirilă
chirila@cs.utt.ro

18 iunie 2002

Rezumat

Împreună cu creșterea dimensiunilor aplicațiilor software a crescut și importanța calității software-ului. Pentru a urmări calitatea software-ului, tehnologia pentru a specifica și evalua calitatea produsului software este de importanță vitală. Este nevoie de un model care să evalueze calitatea sistemelor software.

În această teză vom prezenta un model de arbore de calitate bazat pe strategii de detectie și un instrument software care implementează modelul și măsoară calitatea sistemelor software industriale realizate în tehnologia programării orientate-obiect. Arborele de calitate este un model alcătuit din strategii de detectie și funcții matematice care are ca scop automatizarea procesului de detectie a calității software-ului.

1 Introducere

Un obiectiv principal al ingineriei software este acela de a îmbunătăți calitatea produselor software. Vom realiza acest lucru printr-un mecanism de măsurare a calității unui sistem software pornind de la codul sursă al acestuia.

2 Arbore de calitate bazat pe strategii de detectie

Ne propunem să automatizăm măsurarea calității software-ului prin intermediul arborelui de calitate bazat pe strategii de detectie.

Abordarea se bazează pe articolul domnului Marinescu [2, Marinescu2001a] care se referă la detecția curențelor de proiectare bazate pe metrice și teza sa de doctorat care are ca temă interpretarea mă sură torilor în sisteme orientate obiect [1, Marinescu2002]. Abordarea are la bază **analiza statică a codului sursă** a sistemului software. Mecanismul de măsurare a calității nu lucrează direct pe codul sursă ci exploatează un metamodel care este obținut pe baza acestuia.

2.1 Calitate, factor de calitate

Abordarea calității este una decompozițională, abordare pe care o găsim și în arborele lui McCall și a lui Boehm. Ea este mai apropiată de ideea lui McCall pentru că păstrăm noțiunea de factor în schimb noțiunea de criteriu se metamorfozează în strategie de detecție. Avantajul acestei metamorfoze aceea că prin eliminarea criteriilor care vor conduce la folosirea unor anume metrice, fără a putea interpreta concludent rezultatele și introducerea strategiilor de detecție vom putea verifica dacă este respectată paradigma programării orientate-obiect.

Formal, un arbore de calitate poate fi descris de relațiile următoare:

$$\begin{aligned}
 \text{Calitate} &:= f_F(\text{Factor}_1, \text{Factor}_2, \text{Factor}_3, \dots, \text{Factor}_n) \\
 \text{Factor}_i &:= f_{SD}(\text{StrategieDeDetecție}_1, \text{StrategieDeDetecție}_2, \\
 &\quad \text{StrategieDeDetecție}_3, \dots, \text{StrategieDeDetecție}_n)
 \end{aligned}$$

2.2 Strategie de detecție

Strategia de detecție a unei curențe poate fi definită în felul următor: [2], [3] expresie cuantificabilă a unei reguli prin care entitățile afectate de curența respectivă să poată fi automat detectate. Pornim de la următoarea definiție formală bazată pe metrice:

$$\begin{aligned}
 S &:= M_1^{O_1} * M_2^{O_2} * \dots * M_n^{O_n} \\
 * &:= \cup \quad | \quad \cap \quad | \quad \setminus
 \end{aligned}$$

Implementarea conceptului de strategie de detecție este ilustrată prin regulile următoare:

```
DetectionStrategy      := StrategyDefinition SymbolsDefinition
```

```
# reguli pentru definirea unei strategii de detecție
```

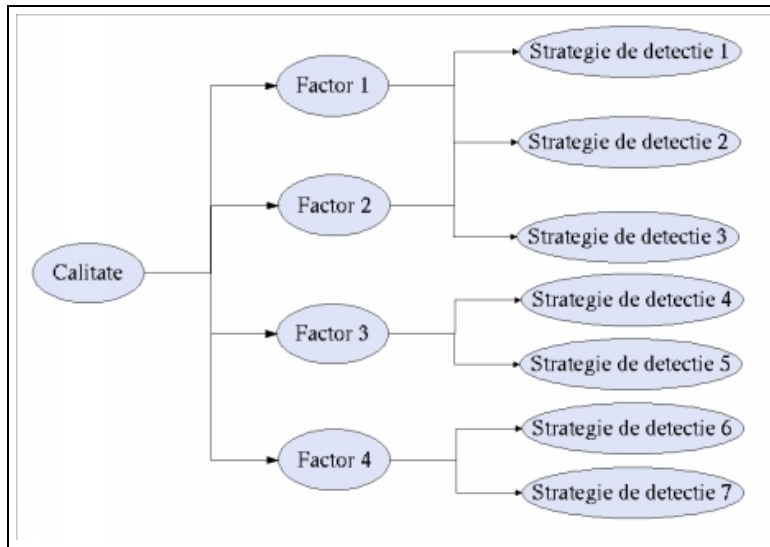


Figura 1: Arbore de calitate bazat pe strategii de detectie - schemă de principiu

```

StrategyDefinition      := StrategyName "==" DetectionRule ";
DetectionRule           := MetricWithOutliers |
                          ComposedDetectionRule
MetricWithOutliers      := "(" MetricName "," OutlierName ")"
ComposedDetectionRule  := DetectionRule CompositionOperator
                          DetectionRule
StrategyName            := [A-z][A-z0-9_]
CompositionOperator     := "or" | "and" | "butnotin"

# simbolurile sunt definitii de metrice si operatori
SymbolsDefinition      := MetricDefinition |
                          OutlierDefinition

# reguli pentru definirea metricilor
MetricDefinition       := MetricName "==" SqlQuery ";"
MetricName             := [A-z][A-z0-9_]
SqlQuery               := [.] # SELECT <Entity> <Value>

# reguli pentru definirea operatorilor statistici
OutlierDefinition     := OutlierName "==" OutlierType
                          "(" OutlierParameter ")" ";"
OutlierType           := "TopValues" | "BottomValues" |
  
```

```

OutlierName      := [A-z] [A-z0-9_]
OutlierParameter := [0-9] [0-9,] [%]

```

O strategie de detecție încapsulează principiile paradigmei programării orientate-obiect cuantificând măsura în care ea este respectată într-un anumit sistem software.

2.3 Operatori și funcții matematice elementare

Funcțiile matematice f_F și f_{SD} folosite în definiția arborelui de calitate au rolul de a lega între ele calitatea, factorii și strategiile de detecție. Din punct de vedere matematic f_F și f_{SD} vor fi funcții compuse din funcții matematice elementare. Putem abstractiza și operatorii aritmetici prin intermediul funcțiilor:

$$\begin{aligned}
 f &= f_F \quad | \quad f_{SD} \\
 f &= f_{e_1} \circ f_{e_2} \circ f_{e_3} \circ \dots \circ f_{e_n} \\
 f_{e_i} &= + \quad | \quad - \quad | \quad * \quad | \quad / \quad | \quad \min \quad | \quad \max \quad | \quad \text{avg}
 \end{aligned}$$

2.4 Tabela de calificative și punctaje

Tabela de calificative și punctaje este un mecanism de măsurare prin în cadrare a gradului de calitate la unul din calificativele disponibile. Formal o tabelă de calificative poate fi descrisă astfel:

$$\begin{aligned}
 \text{Tabela} &= \{ \text{Calificativ}_1, \text{Calificativ}_2, \dots, \text{Calificativ}_n \} \\
 \text{Calificativ}_i &= (\text{LimitaInferioara}, \text{LimitaSuperioara}, \text{PunctajAcordat})
 \end{aligned}$$

Fiecărei trăsături a calității (calitate, factor, strategie de detecție) i se va asocia o astfel de tabelă.

Exemplu de utilizare și interpretare Considerăm următoarea tabelă de calificative:

```

ScoreMap
{
  EXCELLENT 0 5 10,
  GOOD 5 10 8,
}

```

```
ACCEPTABLE 10 15 6,  
POOR 15 +oo 0  
}
```

Tabela de față conține 4 calificative pe care le interpretăm în felul următor: trăsătura de calitate poate fi excelentă dacă au fost detectate între 0 și 5 entități suspecte. Unui astfel de calificativ se acordă conform tabelii 10 puncte. Celelalte calificative se vor interpreta analog.

2.5 Implementarea modelului de arbore de calitate

Gramatica care implementează conceptul de arbore de calitate bazat pe strategii de detecție o vom nota cu SDQT - *Strategy driven Quality Tree*. Setul de reguli după care se poate construi un arbore de calitate se poate descompune în trei mari secțiuni după cele 3 trăsături de calitate considerate:

- reguli referitoare la calitate
- reguli referitoare la factori
- reguli referitoare la strategii.

Pe lângă aceste reguli mai sunt și alte reguli mai de detaliu:

- reguli referitoare la tabela de calificative
- reguli referitoare la parametrizarea strategiilor de detecție
- reguli referitoare la parametrizarea funcțiilor matematice implementate.

Reguli referitoare la trăsăturile de calitate Aceste reguli ne spun cum se compune calitatea pe baza factorilor, sau factorii pe baza strategiilor. Regulile modelează funcțiile compuse f_F și f_{SD} aplicată pe factori respectiv pe strategii de detecție. Sunt modelate reguli care permit crearea de expresii matematice bazate pe operatori aditivi și multiplicativi cu respectarea ordinii execuției operațiilor. Vom analiza o bucată din gramatica SDQT la nivelul definirii calității:

```
# Reguli pentru definirea expresiei calitatii  
...
```

```

03 QualityMathematicalExpressionExpression :=
QualityMathematicalTermExpression
QualityMathematicalExpression2Expression

04 QualityMathematicalExpression2Expression :=
AdditiveOperator QualityMathematicalExpressionExpression | eps

05 QualityMathematicalTermExpression :=
QualityMathematicalFactorExpression
QualityMathematicalTerm2Expression

06 QualityMathematicalTerm2Expression :=
MultiplicativeOperator QualityMathematicalTermExpression | eps

07 QualityMathematicalFactorExpression :=
FactorExpression | Number |
QualityStatisticalFunctionCallExpression |
( QualityMathematicalExpressionExpression )

08 FactorExpression :=
Identifier

09 QualityStatisticalFunctionCallExpression :=
StatisticalFunction ( QualityParameterListExpression )
...

```

În regulile listate mai sus putem remarca expresia calității bazată pe termeni, termenii reprezentând factori, factorii care pot fi apeluri de funcții matematice. Aceleași reguli apar și la celelalte două nivele dar cu mici diferențe sensibile.

2.6 Exemplu de arbore de calitate

Arborele de calitate propus spre studiu este creionat în figura 2:

Descrierea în gramatica SDQT unui astfel de arbore este prezentă pe liniile următoare:

```

# QualityTree-2.0-sample
# 09.06.2002

```

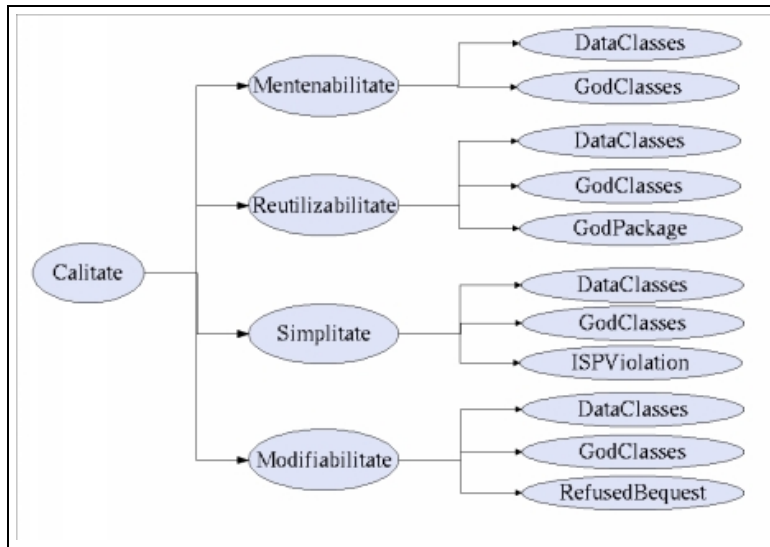


Figura 2: Arbore de calitate bazat pe strategii de detecție - exemplu

```

# Quality Section
Quality :=
avg(Maintainability, Reusability,
    Understandability, Modifiability)
{
    EXCELLENT 8 10 10,
    GOOD 6 6 8,
    FAIR 4 6 6,
    POOR 0 4 4
};

# Factor Section
Maintainability :=
avg(DataClasses(ScoreMap1, (0,0,0)),
    GodClasses(ScoreMap1, (0,0,0)))
{
    EXCELLENT 8 10 9,
    GOOD 6 8 7,
    ACCEPTABLE 4 6 5,
    POOR 0 4 2
},

Reusability :=

```

```

avg(DataClasses(ScoreMap2,(0,0,0)),
     GodClasses(ScoreMap2,(0,0,0)),
     GodPackage(ScoreMap2,(0,0,0)))
{
  EXCELLENT 8 10 9,
  GOOD 6 8 7,
  ACCEPTABLE 4 6 5,
  POOR 0 4 2
},

```

```

Understandability :=
avg(DataClasses(ScoreMap1,(0,0,0)),
     GodClasses(ScoreMap1,(0,0,0)),
     ISPViolation(ScoreMap2,(0,0,0)))
{
  EXCELLENT 8 10 9,
  GOOD 6 8 7,
  ACCEPTABLE 4 6 5,
  POOR 0 4 2
},

```

```

Modifiability :=
avg(DataClasses(ScoreMap1,(0,0,0)),
     GodClasses(ScoreMap2,(0,0,0)),
     RefusedBequest(ScoreMap2,(0,0,0)))
{
  EXCELLENT 8 10 9,
  GOOD 6 8 7,
  ACCEPTABLE 4 6 5,
  POOR 0 4 2
};

```

Strategy ScoreMap Section

```

ScoreMap1
{
  EXCELLENT 0 5 10,
  GOOD 5 10 8,
  ACCEPTABLE 10 15 6,
  POOR 15 +oo 0
},

```



```

ScoreMap2
{
    EXCELLENT 0 5 9,
    GOOD 5 10 7,
    ACCEPTABLE 10 15 5,
    POOR 15 +oo 1
};

```

Vom descrie puțin semantica acestui arbore de calitate simplificat justificând alegerile făcute pentru strategiile de detecție.

Pentru început vom analiza *mentenabilitatea*. Mentenabilitatea este factor de calitate care se referă la toate atributele unui sistem software de care depinde efortul de a face modificări specificate [5]. În acest sens am optat pentru două strategii de detecție: DataClasses și GodClasses, care descoperă nerespectarea principiului paradigmei orientate-obiect conform căreia atributele și comportamentul unui obiect trebuie declarate în aceeași clasă.

Reutilizabilitatea este factor definit prin atributele unui sistem software care se referă la efortul de a reutiliza părți din acesta. Pentru a atinge acest aspect vom apela din nou la cele două strategii DataClasses și GodClasses, dar vom mai opta și pentru strategia GodPackage care detectează pachetele mari în care locuiesc foarte multe clase, și care sunt foarte intens utilizate de clienții acestora. Strategia în capsulează principiul conform căruia clasele care au legături logice între ele sau sunt în relație trebuie împachetate împreună. Evident că reutilizabilitatea este afectată de o astfel de încălcare a principiului enunțat anterior.

Facilitatea de înțelegere a codului poate fi considerată a fi influențată de separarea comportamentului de date la nivelul claselor dar și de nerespectarea principiului segregării interfețelor. Conform acestui principiu pentru fiecare client al unui obiect va exista o interfață prin care clientul va avea acces la serviciile obiectului. Un obiect cu multiple funcționalități va putea fi folosit cu restricție în funcție de context.

Modifiabilitatea unui sistem software o reprezintă setul de atribute pe care se bazează efortul de a aduce modificări de funcționalitate sistemului. Am propus utilizarea strategiilor: DataClasses, GodClasses, RefusedBequest. Ultima strategie detectează clasele care au o relație de moștenire forțată în urma necesității de a adăuga comportament sau stare unui obiect. Relația de moștenire ar putea fi înlocuită cu o relație de compoziție nemaifiind astfel afectată modifiabilitatea sistemului.

3 Concluzii

Utilizând instrumentul software câștigăm:

- *abordare sistematică* prin descrierea modelului de calitate în limbajul SDQT
- *repetabilitate* bazată pe faptul că arborele de calitate poate fi reutilizat, parametrii strategiilor pot varia, dar arborele rămâne aceeași
- *scalabilitate* asigurată de instrumentul software - PRODEOOS cât și de rezultatele obținute pe sisteme industriale.

Bibliografie

- [1] R. Marinescu *Measurements Interpretation in Object-Oriented Systems A Goal-Driven Approach*. Ph. D Thesis, “Politehnica” University Timișoara, 2002 (va fi publicată).
- [2] R. Marinescu. *Detecting Design Flaws via Metrics in Object-Oriented Systems*. Proceedings of the TOOLS-USA 39, ISBN 0-7695-1251-8, IEEE Computer Society, 2001.
- [3] R. Marinescu. *Design Flaws and Detection Strategies*. PhD Presentation, Karlsruhe, April 2001.
- [4] C.B. Chirilă. *Automatizarea detecției curențelor de proiectare în sisteme orientate-obiect*. “Politehnica” University Timișoara 2001.
- [5] ISO/IEC 9126 *Information technology - Software product evaluation - Quality characteristics and their guidelines for their use*
- [6] N.E. Fenton, S.L. Pfleeger. *Software Metrics, A Rigorous & Practical Approach*. PWS Publishing Company, Boston, 1997.
- [7] R. Martin. *Design Principles and Patterns*. <http://www.objectmentor.com>, 2000.