Metryki MOOD w Rational Rose[1]

Ilona Bluemke, Piotr Zając
Instytut Informatyki, Politechnika Warszawska
I.Bluemke@ii.pw.edu.pl

Streszczenie

Pewne metryki oprogramowania obiektowego pozwalają na określenie złożoności projektu już we wczesnych fazach rozwoju. Na podstawie obliczonych metryk można także wnioskować o stopniu trudności testowania oraz identyfikować fragmenty projektu, które powinny być zmienione, by poprawić jakość oprogramowania. Autorzy przedstawiają program Ometrics obliczający wybrane metryki obiektowe dotyczące klas lub całego projektu i zintegrowany z systemem Rational Rose. Zaproponowane skrypty obliczające metryki mogą być użyte w systemie Rose niezależnie od języka implementacji. Pokazano prosty przykład ilustrujący obliczanie metryk MOOD w projekcie realizowanym w Rose.

1. Wprowadzenie

Stosowanie metryk w procesie tworzenia oprogramowania jest już powszechnie akceptowane i wykorzystywane. Popularność jaką zdobyło w ostatnich latach programowanie obiektowe spowodowało, że dokonano wielu prób zdefiniowania oraz użycia metryk dla tego rodzaju oprogramowania. Zaproponowano bardzo wiele specjalizowanych metryk obiektowych, niektóre z nich są dostosowane do konkretnego języka obiektowego, inne można stosować niezależnie od używanego języka implementacji. Przegląd wielu metryk można znaleźć w pracy [BLUE2001], a na stronach dec.bournemouth.ac.uk/ESERG/ bibliography.html można znaleźć streszczenia ponad 500 prac dotyczących metryk obiektowych i opublikowanych w okresie 1986-1999. Metryki mogą dotyczyć pojedynczej klasy lub całego projektu. Najczęściej obecnie stosowane metryki stosowane w fazie projektowania oprogramowania obiektowego zostały zaproponowane przez Chidambera i Kemerera [CHKE1994] (metryki dla klasy) oraz metryki MOOD (Abreu, Goualo i Esteves) [ABGE1995] (metryki dla całego projektu). Metryki umożliwiają znalezienie w projekcie miejsc bardziej złożonych, na których będzie trzeba skoncentrować proces testowania lub które wymagają przeprojektowania. Jako miejsca bardziej złożone systemu należy rozumieć miejsca, gdzie występują klasy o dużej liczbie metod, miejsca gdzie występuje duże zagęszczenie przesyłanych komunikatów lub też miejsce, gdzie z usług jednej klasy korzysta dużo innych klas np. w przypadku głębokiego drzewa dziedziczenia. Metryki mogą pomóc w tworzeniu opro-

[1]Praca finansowana z grantu Dziekana Wydziału Elektroniki i Technik Informacyjnych PW nr 503/G/1032/2510/000

gramowania wysokiej jakości, redukując koszty implementacji, testowania i utrzymania go w czasie całego cyklu życia. Celem autorów jest przedstawienie możliwości obliczania wartości metryk (niezależnych od języka implementacji) w narzędziu CASE - systemie Rational Rose. Istniejące na rynku narzędzia CASE do projektowania obiektowego nie pozwalają na obliczanie metryk. Niektóre narzędzia CASE udostępniają język skryptów, za pomocą którego możliwy jest dostęp do danych dotyczących projektu, a tym samym wyliczenie wartości metryk tworzonego oprogramowania. Do grupy takich narzędzi należy Paradigm [BLZA2000], Select [NOBL2001] czy Rose.

W podrozdziale drugim omówiono w skrócie metryki zaproponowane przez Abreu Brito, Goualo i Esteves [ABGE1995], w rozdziale trzecim przedstawiono ogólne informacje o programie Ometrics obliczania metryk dotyczących klasy (metryki Chidambera i Kemerera i inne) i całego projektu (MOOD). W kolejnym rozdziale przedstawiono przykład. Dodatek A zawiera tekst źródłowy wybranej metryki.

2. Metryki MOOD

Abreu Brito F., Goualo M., Esteves R J. [ABGE1995] zaproponowali zbiór sześciu metryk dla projektu obiektowego *(metrics for object-oriented design*) MOOD:

1. MHF - *Method Hiding Factor* - współczynnik ukrycia metod,
2. AHF - *Attribute Hiding Factor* - współczynnik ukrycia atrybutów,
3. MIF - *Method Inheritance Factor* - współczynnik dziedziczenia metod,
4. AIF - *Attribute Inheritance Factor* - współczynnik dziedziczenia atrybutów,
5. CF - *Coupling Fac*tor – współczynnik sprzężenia,
6. PF - *Polymorphism Factor* - współczynnik polimorfizmu.

Dyskusję dotyczącą poprawności tych metryk można znaleźć w [HACO1998].

2.1. Pomiar hermetyzacji

Metryki MHF (The Method Hiding Factor) oraz AHF (Attribute Hiding Factor) zostały zaproponowane wspólnie w celu pomiaru hermetyzacji i dlatego również wspólnie powinny być rozważane. Metryka MHF zdefiniowana jest równaniem 1.

$$MHF = \frac{\sum_{i=1}^{TC} \sum_{m=1}^{M_d(C_i)} (1 - V(M_{mi}))}{\sum_{i=1}^{TC} M_d(C_i)}$$

(1)

gdzie $M_d(C_i)$ jest liczbą metod zadeklarowanych w klasie *i-tej*, TC jest całkowitą liczbą klas a $V(M_m)$ wyrażono równaniem 2.

$$V(M_m) = \sum_{j=1}^{TC} \frac{is\_visible(M_{mi}, C_j)}{TC - 1}$$

(2)

natomiast *is_visible*(M_m, C_j) pokazano w równaniu 3:

$$is\_visible(M_m, C_j) = \begin{cases} 1 \; gdy \; j \neq i \wedge C_j \; może \; wywołać M_{mi} \\ 0 \; w \; przeciwnym \; przypadku \end{cases}$$

(3)

W związku z powyższym, dla wszystkich klas C_1, C_2, ... C_n metoda liczona jest jako zero, jeżeli może zostać użyta przez inną klasę, a jeden - w przeciwnym przypadku. Całkowita liczba klas w systemie, jest dzielona przez całkowitą liczbę metod zdefiniowanych w systemie, aby obliczyć procent ukrytych metod.

AHF została zdefiniowana w podobny sposób lecz używając atrybuty, a nie metody. Należy zauważyć, że definicje MHF i AHF powodują nieciągłości dla systemów z jedną klasą. Dla systemów napisanych w C++, obliczanie metryki MHF jest utrudnione przez istnienie metod typu *protected*. Dla tych metod w C++, metoda jest liczona jako część ułamkowa pomiędzy 0 a 1 w sposób pokazany w wyrażeniu 4.

$$\frac{liczba \; klas \; nie \; dziedziczących \; metody}{całkowita \; liczba \; klas - 1}$$

(4)

2.2. Pomiar dziedziczenia

Metryka MIF *(Method Inheritance Factor)* – współczynnik odziedziczonych metod zdefiniowana jest równaniem 5.

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_j)}$$

(5)

gdzie: $M_a(C_i) = M_d(C_i) + M_i(C_i)$ oraz:

$M_d(C_i)$ = liczba zadeklarowanych w klasie metod;

$M_a(C_i)$ = liczba metod, która może być wywołana poprzez asocjację z klasą C_i

$M_i(C_i)$ = liczba metod dziedziczonych w klasie C_i (a nie przeciążonych).

Dla metryki MIF, dla każdej klasy C_1, C_2, ... C_n metoda jest liczona jako 0, jeśli nie została odziedziczona, oraz jako 1 jeśli została odziedziczona. Suma dla całego systemu jest dzielona przez całkowitą liczbę metod, wliczając metody odziedziczone (np. metody, które są dziedziczone są liczone jako należące do ich bazowych klas, jak również do wszystkich dziedziczących podklas). Metryka AIF *(Attribute Inheritance Factor)* jest zdefiniowana analogicznie. Oznacza to, że metryki MIF i AIF mierzą liczbę dziedziczonych metod i atrybutów, jako proporcję całkowitej liczby metod (MIF) czy atrybutów (AIF).

2.3. Pomiar sprzężenia

Metryka CF (*Coupling Factor*)- współczynnik sprzeżenia została zaproponowana do pomiaru sprzężenia pomiędzy klasami, wykluczając sprzężenia poprzez dziedziczenie. Metryka CF jest zdefiniowana równaniem 5.

$$CF = \frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$

(5)

gdzie $is\_client(C_C, C_S)$ pokazano wyrażeniem 6.

$$is\_client(C_C, C_S) = \begin{cases} 1 \text{ gdy } C_C \Rightarrow C_S \wedge C_C \neq C_S \\ 0 \text{ w przeciwnym przypadku.} \end{cases}$$

(6)

a $C_C \Rightarrow C_S$ reprezentuje relację pomiędzy klasą C_C a świadczącą usługi klasą C_S.

Metryka CF jest liczona poprzez rozważenie wszystkich możliwych par klas oraz stwierdzenie, czy klasy w danej parze są sprzężone; poprzez przesyłanie informacji lub też poprzez asocjacje (czyli odwołanie się przez jedną klasę do atrybutu lub metody drugiej klasy). Te relacje są uważane jako równoważne dopóki dotyczą sprzężenia. A zatem CF służy do bezpośredniego pomiaru związku pomiędzy dwoma klasami lub wszystkich par relacji pomiędzy klasami w systemie.

Są dwa możliwe podejścia do obliczania metryki CF. Po pierwsze, można rozważyć metrykę CF jako bezpośredni pomiar sprzężenia pomiędzy klasami. Można również rozważyć metrykę CF jako pośredni pomiar poniższych atrybutów:

- złożoności,
- braków w hermetyzacji,
- braku możliwości ponownego użycia,
- trudności w zrozumieniu,
- trudności związane z utrzymaniem.

W przypadku rozważania metryki CF, jako pomiaru pośredniego przedstawionych powyżej atrybutów, należy odpowiedzieć na pytanie, czy wysoki stopień sprzężenia wskazuje na wysoki stopień złożoności. Prawdopodobnie nie, jako że możliwe jest wykonanie prostego systemu z niewielkim lub żadnym stopniem sprzężenia. Jeżeli rozważy się hermetyzację w celu sprawdzenia zdolności kompilacji modułów oddzielnie, wtedy relacja pomiędzy hermetyzacją, a metryką CF również nie jest klarowna. Dopóki będzie brało się pod uwagę ponowne użycie, będzie niski stopień sprzężenia, ale wysoki stopień ponownego użycia (np. poprzez dziedziczenie). Trudności w zrozumieniu mają związek ze złożonością, co oznacza, że można znaleźć klasę, która nie jest sprzężona, ale trudno zrozumieć zasadę jej działania. Podobnie jest z utrzymaniem. Podsumowując, trudno jest wypowiedzieć się na temat użyteczności metryki CF do pomiaru tych zewnętrznych atrybutów.

2.4. Pomiar polimorfizmu

Metryka PF *(Polymorphism Factor)* – współczynnik polimorfizmu służy do mierzenia potencjalnego polimorfizmu. Metrykę PF wyrażono równaniem 7.

$$PF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

(7)

gdzie:

$$M_d(C_i) = M_n(C_i) + M_o(C_i)$$

(8)

oraz:

$M_n(C_i)$ = liczba nowych metod klasy C_i, które nie przeciążają dziedziczonych;

$M_o(C_i)$ = liczba metod klasy C_i, które przeciążają metody dziedziczone;

$DC(C_i)$ = liczba potomków (liczba klas pochodzących od klasy $M_a(C_i)$.

Metryka PF jest liczbą metod, które przedefiniowują dziedziczone metody, podzieloną przez całkowitą liczbę różnych możliwych sytuacji polimorficznych (reprezentuje to sytuację, gdy wszystkie nowe metody w klasie, są przeciążane we wszystkich klasach pochodnych). A zatem, metryka PF jest pośrednim pomiarem relatywnej liczby dynamicznych związków w systemie. Rozważając zasadność użycia metryki PF jako metryki pośredniej można zauważyć, że mianownik jako mnożnik, zawiera liczbę klas pochodnych klasy bazowej, a więc wartość metryki PF dla systemu bez jakiegokolwiek dziedziczenia zawsze będzie niezdefiniowana. Oznacza to, że metryka ta wykazuje nieciągłość, dając niezdefiniowane rezultaty wtedy, kiedy można by się spodziewać wartości równej zero.

2.5. Analiza metryk MOOD

Wartości metryk dla trzech kolejnych wersji (R1, R2 oraz R3) laboratoryjnego systemu elektronicznego ERS [HACO1998] stworzonego we współpracy pomiędzy Uniwersytetami Southampton i Manchester oraz dla zbioru programów służących do obróbki obrazu (EFOOP2) są pokazane w Tabeli 1. Warta uwagi jest różnica w liczbie metod i atrybutów pomiędzy wersjami R1 i R2 dla systemu ERS. Całkowita liczba metod i atrybutów ustabilizowała się dopiero w wersji R3. Tabela 1 pokazuje wartości metryk MOOD dla trzech wersji (R1, R2, R3) dla systemu ERS oraz systemu EFOOP2.

Tabela 1. Metryki MOOD dla systemu ERS (R1, R2 i R3) oraz system EFOOP2 (Źródło: [HACO1998])

	R1 (%)	R2(%)	R3(%)	EFOOP2(%)
AHF	100	100	100	100
MHF	0	20.4	20.4	6.3

AIF	12.5	0	0	0
MIF	9.1	0	0	0
CF	0	5.8	5.8	3.0
PF	60	Niezdef.	Niezdef.	Niezdef.

Metryka AHF dla wszystkich trzech przypadków ma maksymalną wartość 100% wskazując, że wszystkie atrybuty zostały zadeklarowane jako prywatne. Z drugiej strony, metryka MHF ma relatywnie niską wartość, wskazując braki w ukryciu informacji. Dziedziczenie nie zostało wykorzystane w ogóle, za wyjątkiem systemu ERS-R1, co zostało wykazane przez metrykę MIF oraz AIF. Niezdefiniowane wartości metryki PF również odzwierciedlają brak dziedziczenia w innych systemach. Wszystkie systemy wykazywały jedynie małą wielkość sprzężenia między klasami (wykazanymi przez metrykę CF) wskazując prawdopodobnie na dobrze zaprojektowane systemy.

W tabeli 2. przedstawiono wartości metryk MOOD kolejno dla dziewięciu przykładów komercyjnych systemów z rozmiarami sięgającymi od 16000 do 47000 linii kodu (LOC). Można zauważyć, że wartości dla metryki AHF wahają się pomiędzy 44% a 68%. Sugeruje to równowagę pomiędzy atrybutami publicznymi i prywatnymi. Wartość tej metryki różni się od wysokich wartości procentowych dla omawianych wcześniej systemów ERS i EFOOP2, w których wszystkie atrybuty zostały zadeklarowane jako prywatne. W idealnym przypadku metryka AHF powinna mieć wartość zbliżoną do 100%, co oznacza, że wszystkie atrybuty zostaną zadeklarowane jako prywatne.

Wartości metryki MHF wahają się pomiędzy 8% a 25%. Tak niskie wartości, wskazują na niski stopień ukrycia informacji.

Metryka AIF oscyluje w pobliżu 47%. Jest to wartość raczej niska, co wskazuje na niewielkie użycie dziedziczenia. Podobne wyniki zostały uzyskane dla metryki MIF (od 14% do 46%). Niskie wartości tych metryk wskazują na brak wykorzystania dziedziczenia.

Tabela 2. Metryki MOOD, dla dziewięciu systemów komercyjnych (Źródło: [HACO1998])

	1 %	**2 %**	**3 %**	**4 %**	**5 %**	**6 %**	**7 %**	**8 %**	**9 %**
AHF	45.9	66.7	66.3	44.0	62.5	67.5	52,4	48,5	50,8
MHF	10.1	7.7	16.4	9.5	25.4	15.4	15.8	15.7	15.4
AIF	17.1	11.3	15.3	30.6	46.8	26.1	19.7	36.6	32.0
MIF	15.2	14.3	20.7	27.4	45.5	33.6	22.5	36.5	26.5
CF	3.5	3.5	3.8	6.3	3.1	4.5	5.4	4.9	4.6
PF	4.3	5.4	8.9	2.9	6.7	4.5	6.6	6.2	6.4

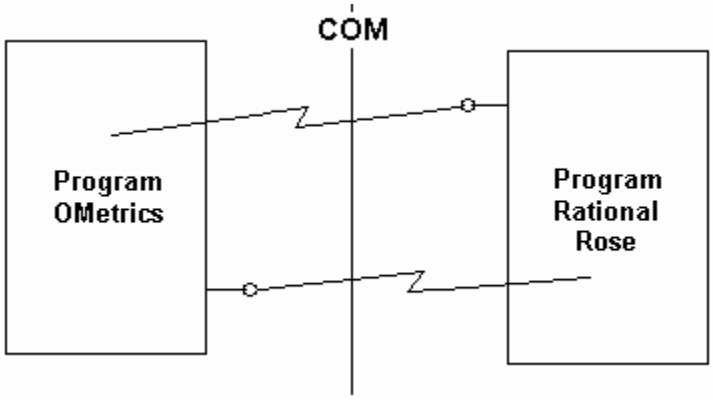
Wartości procentowe dla współczynnika sprzężenia CF wahają się od 3% do 6% co wskazuje na niewielkie sprzężenie pomiędzy klasami. Abreu [ABGE1995] sugeruje, że wartość metryki CF nie powinna być ani zbyt duża ani zbyt mała. Niska wartość sprzężenia redukuje potencjalnie szkodliwe działanie uboczne niepotrzebnych zależności między klasami i ograniczonego ponownego użycia. Bardzo niska wartość metryki CF (0%) wskazuje, że w systemie nie ma sprzężeń pomiędzy klasami, co może prowadzić do systemów, w których klasy komunikują się wzajemnie jedynie poprzez dziedziczenie, lub w których istnieje nadmierna duplikacja kodu.

Z drugiej strony 100% wartość metryki CF może również wskazywać na problematyczną strukturę komunikacyjną; zbytnie sprzężenie oznacza, że oprogramowanie będzie trudne w utrzymaniu i ponownym użyciu. Powyższe wartości dla metryki PF rozciągają się pomiędzy wartościami 3% a 9%; tak niskie wartości są typowe i biorą pod uwagę relatywnie niskie użycie dziedziczenia.

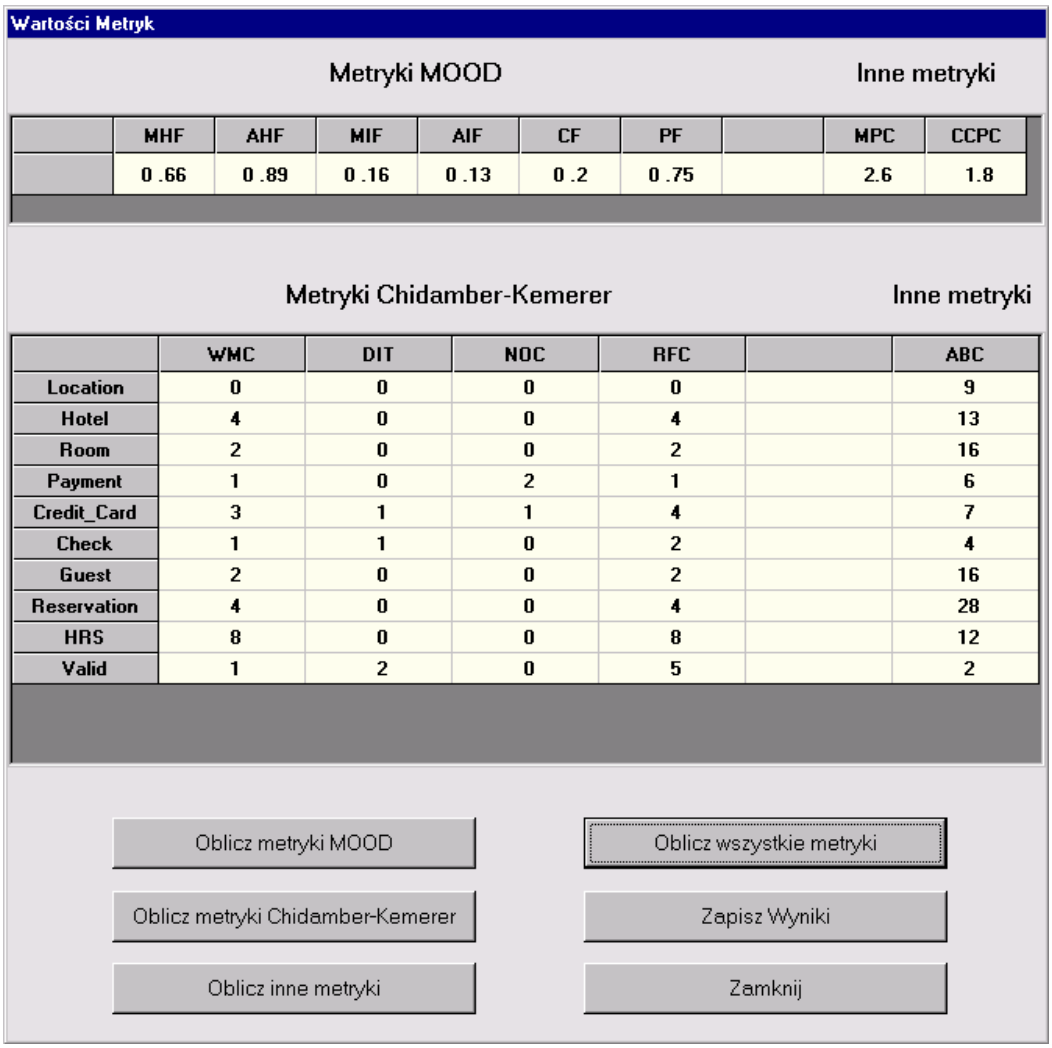
3. System OMetrics

Program OMetrics oblicza wybrane metryki obiektowe na podstawie informacji znajdujących się na diagramach klas modelu- projektu stworzonego w Rose. Spośród wielu znanych metryk obiektowych do realizacji zostały wybrane metryki klasy Chidambera i Kemerera, a dla całego projektu metryki MOOD. Fragmenty kodu wykorzystywane do obliczania metryk mają postać skryptów Rational Rose, które za pomocą interfejsów COM (Component Object Model) komunikują się z tą aplikacją w celu pobrania danych potrzebnych do dokonania obliczeń.

Po wybraniu modelu Rational Rose, przy pomocy interfejsów COM (rysunek 1) wywoływana jest instancja aplikacji Rational Rose, otwierany jest wybrany poprzednio model, a Rational Rose przechodzi w stan oczekiwania (okno aplikacji nie jest widoczne na pasku zadań). Od tego momentu można swobodnie operować na obiektach otwartego modelu. Oznacza to, że istnieje możliwość dostępu do wszystkich klas oraz relacji znajdujących się w otwartym modelu. W każdej chwili można dokonać wyboru innego modelu, bez konieczności ponownej inicjalizacji programu Rational Rose.



Rys. 1. Współpraca OMetrics i Rose



Rys. 2. Okno systemu OMetrics

W kodzie aplikacji zaszyty jest zbiór 12 skryptów Rational Rose (przykład jednego skryptu zawiera Dodatek A), służących do obliczania metryk obiektowych.

Istnieje możliwość wyboru do obliczeń pojedynczej metryki, lub też poszczególnych zbiorów metryk np. Chidamber-Kemerer, MOOD, inne. Rezultatem wyboru odpowiedniej metryki jest wykonanie związanego z nią skryptu na wybranym poprzednio modelu- projekcie Rational Rose. Jeżeli dokonamy wyboru zbioru metryk (rysunek 2) to nastąpi wykonanie wszystkich związanych z nimi skryptów.

Wynikiem działania skryptów są przedstawiane na ekranie wartości. Istnieje także możliwość zapisania wyników obliczeń do pliku tekstowego. Dokładniejszy opis systemu zawarto w pracy [ZAJĄ2003].

4. Przykład

Na rysunku 3. przedstawiono przykładowy diagram klas a poniżej obliczone wartości metryk MOOD dla tego projektu.

```
Wartości metryk dla projektu asocjacje.mdl
```

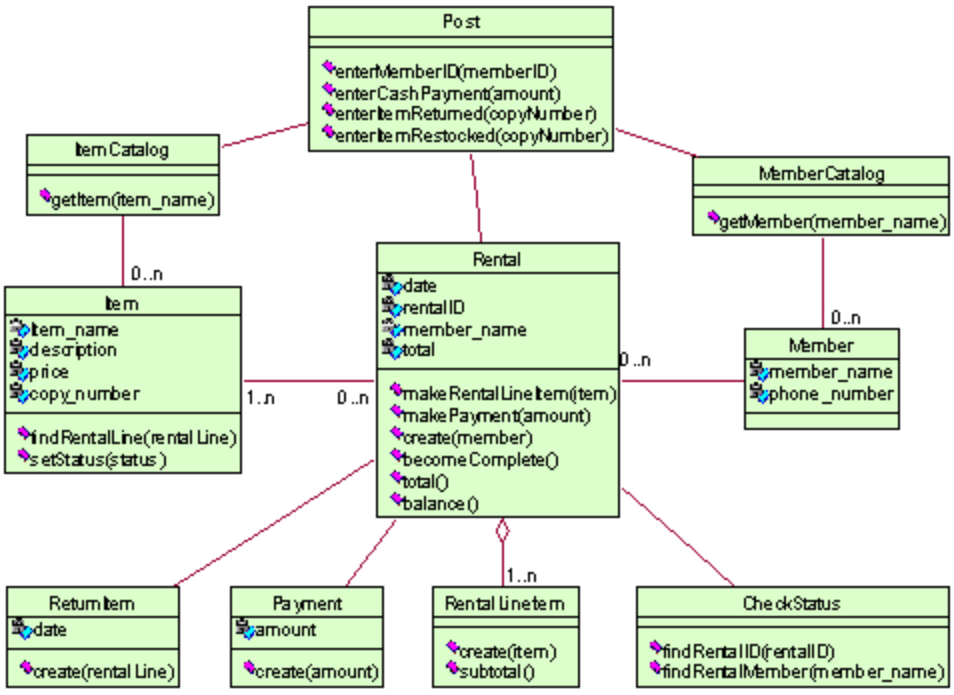
```
-----------------------------------
MHF      - 0.62
AHF      - 1
MIF      - 0
AIF      - 0
CF       - 0.24
PF       - Niezdefin.
```

Wartość metryki MHF określa widoczność publicznych metod poprzez relacje asocjacji. Metryka MHF = 0.62 co oznacza 62% ukrycie metod w systemie. Duży wpływ na taki spadek ukrycia metod w systemie ma klasa `Rental`, która zawiera ponad 30% wszystkich metod i które są widoczne dla 7 innych klas.

Wartość metryki AHF=1 oznacza 100% ukrycie pól w systemie (wszystkie pola zostały zadeklarowane jako prywatne).

Ze względu na brak dziedziczenia, wskaźniki MIF i AIF dziedziczenia metod i pól są równe zero. Wartość metryki CF=0.24 wskazuje na 24% sprzężenie klas w systemie. Wartość ta nie powinna być zbyt wysoka, gdyż wskazywałoby to na dużą liczbę relacji asocjacji.

W trakcie obliczeń wartości metryki PF, liczbę metod przeciążonych dzieli się przez iloczyn liczby metod nie przeciążonych i liczby potomków danej klasy. Ponieważ na diagramie nie ma relacji generalizacji, dlatego mianownik jest równy 0. Z tego względu metryka PF ma wartość niezdefiniowaną dla diagramu z rysunku 3.



Rys. 3. Przykładowy diagram klas

5. Podsumowanie

W Instytucie Informatyki PW zrealizowano skrypty pozwalające na obliczanie metryk MOOD oraz metryk zaproponowanych przez Chidamber i Kemerer [CHKE1994] dla projektów realizowanych w Rational Rose [ZAJĄ2003].

Metryki MOOD operują na poziomie systemowym. Porównując je z metrykami zaproponowanymi przez Chidamber i Kemerer [CHKE1994, BLUE2001, BLZA2000] można zauważyć, że te dwa zbiory dopełniają się, oferując różne oszacowania systemów. Metryki zaproponowane przez Chidamber i Kemerer wydają się być użyteczne dla projektantów i deweloperów systemu, dając im możliwość pomiaru systemu na poziomie klas. Metryki MOOD mogą być użyteczne dla kierowników projektów, jako metryki operujące na poziomie systemu dostarczają pewnych oszacowań systemu. Zastosowanie praktyczne metryk MOOD wymaga wykonania pomiarów istniejących systemów i na ich podstawie wyprowadzenie wniosków o „dobrych", pożądanych wartościach metryk w „dobrych" systemach. Metryki MOOD zostały zaproponowane w 1995 roku jednak do dnia dzisiejszego, według wiedzy autorów, niewiele jest prac omawiających ich praktyczną przydatność, wykonano niewiele eksperymentów. Pewne informacje na ten temat można znaleźć w [HACO1998], [ABCU1998] oraz w [ZHTO2001] (praca nie jest napisana w języku angielskim).

Dodatek A - Metryka MHF

Poniżej przedstawiono skrypt obliczający wartość metryki MHF.

```
Public Sub MHF(Zbior As Boolean)
    Dim i As Integer
    Dim j As Integer
    Dim m As Integer
    Dim a As Integer
    Dim MdCi As Integer
    Dim VMmi As Integer
    Dim tekst As String
    Dim SumVisible As Double
    Dim SumMethods As Double
    Dim SumClasses As Double
    Dim Result As Double
    Dim IsVisible As Integer   ' czy metoda jest widoczna z innej klasy
    Dim MHF As Double
    Dim myRoseApp As RoseApplication
    Set myRoseApp = myRoseAppZbior
    Dim myModel As RoseModel
    Set myModel = myRoseApp.CurrentModel
    Dim aClass As RoseClass
    Dim aClass1 As RoseClass
    Dim aClass2 As RoseClass
    Dim metody As Integer
    Dim aOperation As RoseOperation
    Dim aAssociations As RoseAssociationCollection
    Dim aAssociation As RoseAssociation
    Dim theClasses As RoseClassCollection
```

```
    Dim theClasses1 As RoseClassCollection
    Dim theClasses2 As RoseClassCollection
    Dim theSubClasses As RoseClassCollection
    Dim theGeneralizations As RoseInheritRelationCollection
    Dim theGeneralization As RoseInheritRelation
    Set theClasses = myModel.GetAllClasses()
    Set theClasses1 = myModel.GetAllClasses()
        Classes = theClasses.Count
    metody = 0
    For i = 1 To Classes
        Set aClass = theClasses.GetAt(i)
        metody = metody + aClass.Operations.Count
    Next i
    SumClasses = 0
    For i = 1 To Classes
        Set aClass1 = theClasses1.GetAt(i)
        MdCi = aClass1.Operations.Count
        SumMethods = 0
        For m = 1 To MdCi
            Set theClasses2 = myModel.GetAllClasses()
            Set aOperation = aClass1.Operations.GetAt(m)
            SumVisible = 0
            For j = 1 To Classes
                IsVisible = 0
                Set aClass2 = theClasses2.GetAt(j)
                tekst = aClass1.Name
                tekst = aClass2.Name
                If aClass1.GetUniqueID = aClass2.GetUniqueID Then
                    IsVisible = 0
                ElseIf aOperation.ExportControl.Name = "PrivateAccess" Then
                    IsVisible = 0
                Else
                    Set aAssociations = aClass2.GetAssociations
                    For a = 1 To aAssociations.Count
                        Set aAssociation = aAssociations.GetAt(a)
                        tekst = aAssociation.Role1.Class.Name
                        tekst = aClass1.Name
                        If aAssociation.Role1.Class.GetUniqueID = _
                  aClass2.GetUniqueID Then
                            If aAssociation.Role2.Class.GetUniqueID = _
     aClass1.GetUniqueID Then
                                IsVisible = 1
                                Exit For
                            End If
                        Else
                            If aAssociation.Role1.Class.GetUniqueID = _
     aClass1.GetUniqueID Then
                                IsVisible = 1
                                Exit For
                            End If
                        End If
                    Next a
                End If
                SumVisible = SumVisible + (IsVisible / (Classes - 1))
            Next j
            SumMethods = SumMethods + (1 - SumVisible)
        Next m
```

```
        SumClasses = SumClasses + SumMethods
    Next i
    Result = Round(SumClasses / metody, 2)
    tekst = "Metryka MHF dla modelu  " + Projekt + Chr(13) + Chr(10)
    tekst = tekst + "-------------------------------------------------_
    --------------------" + Chr(13) + Chr(10)
    tekst = tekst + "METRYKA MHF =  :  " + Str(Result)
End Sub
```

Bibliografia

[ABGE1995] F. Abreu Brito, F. Goualo i F. Esteves, *Toward design quality evaluation of object oriented software systems*, Proceeding of the 5th International Conference on Software Quality, Austin, Texas, USA, 1995.

[ABCU1998] F. Abreu Brito i F. Cuche, *Collecting and analyzing the MOOD2 metrics*, Proceeding of the Workshop Object-Oriented Product Metrics for Software Quality Assessment, Brusels, 21 July 1998, ECOOP'98.

[BLUE2001] I. Bluemke, *Metrics for Assessing Complexity and Testability of Object –Oriented Software*, Prodialog, nr 13, s 1-13, 2001.

[BLZA2000] I. Bluemke i I. Zając, *Metryki obiektowe w systemie Paradigm*, Materiały II Krajowej Konferencji Inżynierii Oprogramowania, 18-29 Październik, Zakopane, s. 147-154, 2000.

[CHKE1994] S. R. Chidamber i S. R. Kemerer, *A metrics suite for object oriented design*, IEEE Transactions on Software Engineering, vol 20, no 6, June, pp 476-492, 1994.

[HACO1998] R. Harrison i R. Counsell, *An evaluation of the MOOD set of object-oriented software metrics*, IEEE Transactions on Software Engineering, vol 24, 6, June, 1998, pp. 491-495.

[NOBL2001] S. Nowak i S. Bluemke, *Środowisko wspomagające testowanie oprogramowania obiektowego*, Materiały III Krajowej Konferencji Inżynierii Oprogramowania, 17-20 Październik, Otwock, s. 427- 436, 2001.

[Zają2003] P. Zając, *obiektowe w systemie Rose*, Praca dyplomowa, Instytut Informatyki Politechniki Warszawskiej, 2003.

[ZHTO2001] L. Zhong, L. Tong i L. Zou, *Object-oriented metrics MOOD set and it's application analysis*, Minimicro Systems, vol 22, no 3, pp. 342-344, GAI-KAN BIAJIBU 2001.