

Wykład 5

Charakterystyki zewnętrzne a wewnętrzne oprogramowania (1)

- definicje metryk międzymodułowych i
modułowych

autor: Zofia Kruczkiewicz

Zagadnienia

- 1. Podstawowe definicje dotyczące złożoności strukturalnej oprogramowania**
- 2. Podstawy formalne pomiaru oprogramowania**
- 3. Podstawowe metryki**
- 4. Metryki złożoności międzymodułowej**
- 5. Metryki złożoności modułowej**

1. Podstawowe definicje dotyczące złożoności strukturalnej oprogramowania

Struktura programu to:

- przedstawienie programu na różnych poziomach abstrakcji rozumiane jako odseparowanie danych od bezpośredniej reprezentacji – wynika to z sekwencyjnego przebiegu procesu myślowego i jednocześnie z możliwości wyobrażenia sobie zaledwie ograniczonej liczby pojęć
- podział programu na podsystemy, moduły, klasy, funkcje.

Problem złożoności struktury programu odgrywa kluczową rolę w:

- testowaniu programu, czyli osiągnięciu jak największej jego niezawodności
- rozwijaniu programu wynikającego z możliwości zrozumienia programu i stopnia osiągniętej abstrakcji w dziedzinie danych i operacji
- pielęgnacji programu
- wielokrotnemu zastosowaniu elementów programu (biblioteki, moduły).

Złożoność struktury programu jest odpowiedzialna za następujące *atrybuty (charakterystyki) zewnętrzne oprogramowania:*

1) jakość oprogramowania:

- testowalności, a więc również niezawodności,
- stopnia osiągniętej abstrakcji
- zrozumiałości programu
- stopnia pielęgnacji
- wieloużywalności

2) funkcjonalność

3) koszt.

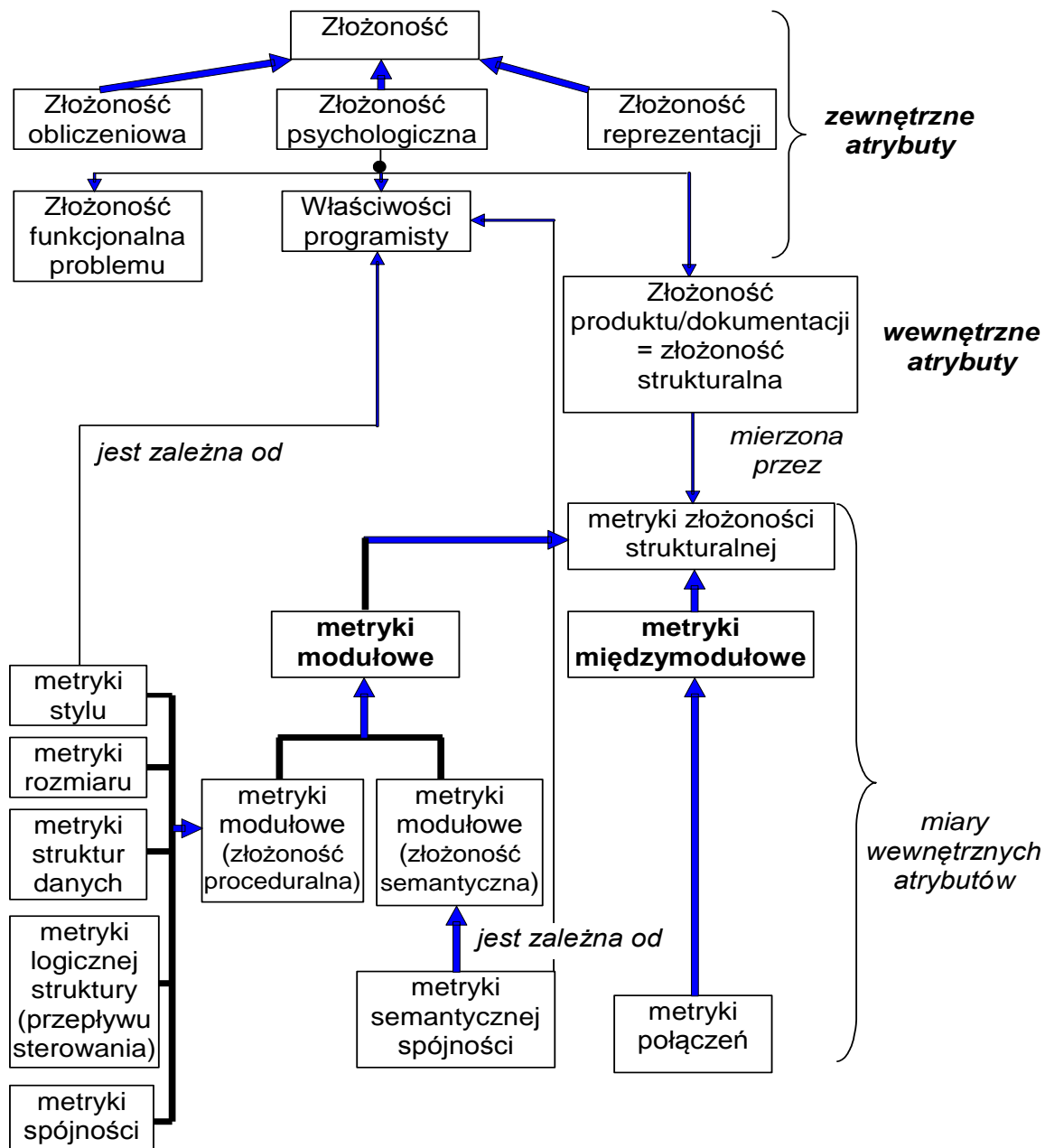
Złożoność struktury programu jest reprezentowana za pomocą **atrybutów (charakterystyk) wewnętrznych oprogramowania.**

Atrybuty wewnętrzne oprogramowania są wyrażane w postaci obiektywnych miar tych atrybutów czyli tzw. **metryk**, czyli prostych wyrażeń, wiążących pewne elementy programu (projektu, kodu źródłowego itp.). Wybór elementów wynika z ich odpowiedzialności za dany atrybut wewnętrzny, a wyrażenie określa wartościowanie atrybutu.

Wyróżnia się następujące **atrybuty wewnętrzne** oprogramowania:

- **atrybuty międzymodułowe** czyli wszelkie związki między modułami (przekazywanie sterowania i parametrów, wspólne korzystanie z pól danych, przy projektowaniu jednego modułu uwzględnia się właściwości innego modułu)
- **atrybuty modułowe** związane z semantycznymi zależnościami między elementami modułu oraz charakterystykami: stylu programowania, rozmiaru oprogramowania, charakteru struktur danych, przepływu sterowania czyli struktury logicznej oprogramowania, oraz spójności oprogramowania jako związku między funkcjami działającymi na danych, a tymi danymi.

Klasyfikacja złożoności oprogramowania - związki między zewnętrznymi atrybutami oprogramowania i miarami wewnętrznych atrybutów (wg Brian Henderson-Sellers)



2. Podstawy formalne pomiaru oprogramowania

„**Pomiar** jest to proces, w którym atrybutom elementów świata rzeczywistego przydzielane są liczby lub symbole w taki sposób, aby charakteryzować te atrybuty według określonych zasad.

Jednostki przydzielane atrybutom w ramach pomiaru nazywane są **miarą** danego atrybutu.

Metryka to proponowana (postulowana) miara.” [15]

Formalna definicja elementów środowiska pomiarowego metryk [12], m. in.:

- system relacji:
- $A = (A, R_j, o_j)$, gdzie A oznacza niepusty zbiór obiektów rzeczywistych z dziedziny oprogramowania, R_j są empirycznymi relacjami (np. równy, większy), o_j oznacza binarne operacje w zbiorze A ;
- $B = (B, S_j, o_j)$, gdzie B jest niepustym zbiorem formalnych obiektów (np. liczby, wektory) w dziedzinie pomiarów, S_j są relacjami (np. równy, większy) w zbiorze B , o_j oznacza binarne operacje w zbiorze B ;
- odwzorowanie homomorficzne: $\mu: A \rightarrow B$, stąd mamy $\forall a \in A \exists b \in B (\mu(a) = b)$;
- typy skal jako (A, B, μ) : nominalna, porządkowa, przedziałowa, absolutna, względna.

Typ skali ogranicza obszar odwzorowań, czyli ogranicza pomiar.

Przykłady skal:

- a) skala nominalna: przypisanie obiektom pewnych ustalonych etykiet
np. numery autobusów
- b) skala porządkowa: przypisanie obiektom etykiet wg ustalonego porządku
np. false, true
- c) skala przedziałowa: znaczenie ma odległość między obiektami:
 $g(x)=a*x+b$ ($a>0$)
np. temperatura w stopniach Celsjusza
- d) skala względna: znaczenie ma odległość między obiektami oraz jedyny punkt odniesienia: $g(x)=a*x$ ($a>0$)
np. temperatura w stopniach Kelvina, pomiar długości, pomiar czasu
- e) skala absolutna: każdy obiekt ma jedną dopuszczalną wartość $g(x) = x$
np. liczba liter w zdaniu, miary statystyczne (wartość przeciętna, mediana, wariancja itp.)

Aksjomaty oceny złożoności oprogramowania:

1. Przy porównywaniu dwóch programów pomija się fragmenty o tej samej złożoności.
2. Istnieje skończona liczba programów o tej samej złożoności.
3. Istnieją różne programy P i Q, takie że mają identyczną złożoność
4. Istnieją równoważne $P \equiv Q$ tzn. spełniające te same funkcje, lecz o różnej złożoności.
5. Złożoność programów P i Q połączonych jest niemniejsza od złożoności każdego z osobna.
6. Dodanie do każdego z pewnych programów P i Q o identycznej złożoności pewnego programu R może spowodować różnice w złożoności uzyskanych programów.
7. Jeżeli program Q powstał przez permutację porządku elementów programu P, to P i Q mają różną złożoność.
8. Jeśli program Q różni się jedynie nazwami od programu P (czyli Q powstał z P po zmianie nazw), to P i Q mają identyczną złożoność.
9. Złożoność programu w wyniku połączenia dowolnych programów P i Q jest większa niż suma złożoności każdego z nich.

3. Podstawowe metryki

Poszczególne metryki złożoności strukturalnej mogą być wyznaczone w dowolnym stadium rozwoju oprogramowania, jednak to rzutuje na wybór elementów produktu i rodzaj wyrażenia.

Całkowita złożoność C_p programu jest równa:

$$C_p = C_I + C_M + (C_{Hmax} - C_H)$$

gdzie

- C_I -złożoność międzymodułowa,
- C_M -złożoność modułu,
- C_{Hmax} - całkowita złożoność wynikająca ze spójności modułu zmniejszona o złożoność semantyczną modułu C_H

4. Metryki złożoności międzymodułowej

Oslabienie powiązań między-modułowych prowadzi do zmniejszenia oddziaływań między modułami oraz poprawy struktury oprogramowania.

Elementami łączącymi wyjściowymi z innymi modułami są:

- **Funkcja/metoda wywołująca funkcję z innego modułu**
- wszystkie elementy importowane z innych modułów
- każda informacja z poza modułu potrzebna do zdefiniowania ciała funkcji (np. obsługa błędów), definicji typu strukturalnego, definicji dowolnej zmiennej

Elementami łączącymi wejściowymi dany moduł z innymi modułami są:

- **funkcja/metoda danego modułu wywoływana przez funkcję/metodę z innego modułu**
- Wszystkie elementy modułu przekazywane w importowanych modułach
- informacja zawarta w module potrzebna w innych modułach do dowolnej definicji (np. obsługa błędów), definicji typu strukturalnego, definicji dowolnej zmiennej

RFC - metryki połączeń wyjściowych

$$\text{RFC} = M + R \text{ oraz } \text{RFC}' = M + R'$$

Zakres wartości (1 – 50)

gdzie

M – liczba w danej klasie

R – liczba metod wywoływanych przez metody M z innych klas

R' – R + pozostałe metody wywoływane zgodnie z drzewem wywołań

R i R' są wywołanymi metody zwykłymi lub wirtualnymi (tyle razy liczonymi, ile klas przesłania metodę)

Uwagi:

1. Duża wartość metryki oznacza dużo błędów
2. Duża wartość **metryki** oznacza duży wysiłek przy testowaniu
3. Duża wartość metryki oznacza trudność w zrozumieniu klasy

CBO – metryka połączeń wyjściowych z innymi klasami, z którymi jest powiązana dana klasa

Zakres wartości (0..14)

Wartość metryki oznacza liczbę klas powiązanych przez wywołanie metod zwykłej lub wirtualnej innych klas (tyle razy liczonej, ile klas przesłania metodę), zastosowanie odwołań do zmiennej (wzajemne powiązanie między klasami jest liczone tylko raz) własnej klasy i przez dziedziczenie, przez argumenty metody, przez typy danych zwracane przez return oraz powiązania za pomocą wyjątków – wartość do 14

Uwagi:

1. Zbyt duża wartość wymaga dużego wysiłku przy testowaniu
2. Ograniczone zastosowanie zbyt powiązanej klasy w innych programach – gorsza wieloużywalność

Fan-out – metryka połączeń wyjściowych

Metryka *Fan-out* wyznacza liczbę połączeń *elementów wyjściowych* jednego modułu z *elementami wejściowymi* innych modułów. Uwzględnia się tylko jedno dowolne połączenie wyjściowe-wejściowe z każdym z modułów.

Fan-in – metryka połączeń wejściowych

Metryka *Fan-in* wyznacza liczbę połączeń *elementów wejściowych* jednego modułu z *elementami wyjściowymi* innych modułów. Uwzględnia się tylko jedno dowolne wejściowo-wyjściowe połączenie z każdym z modułów.

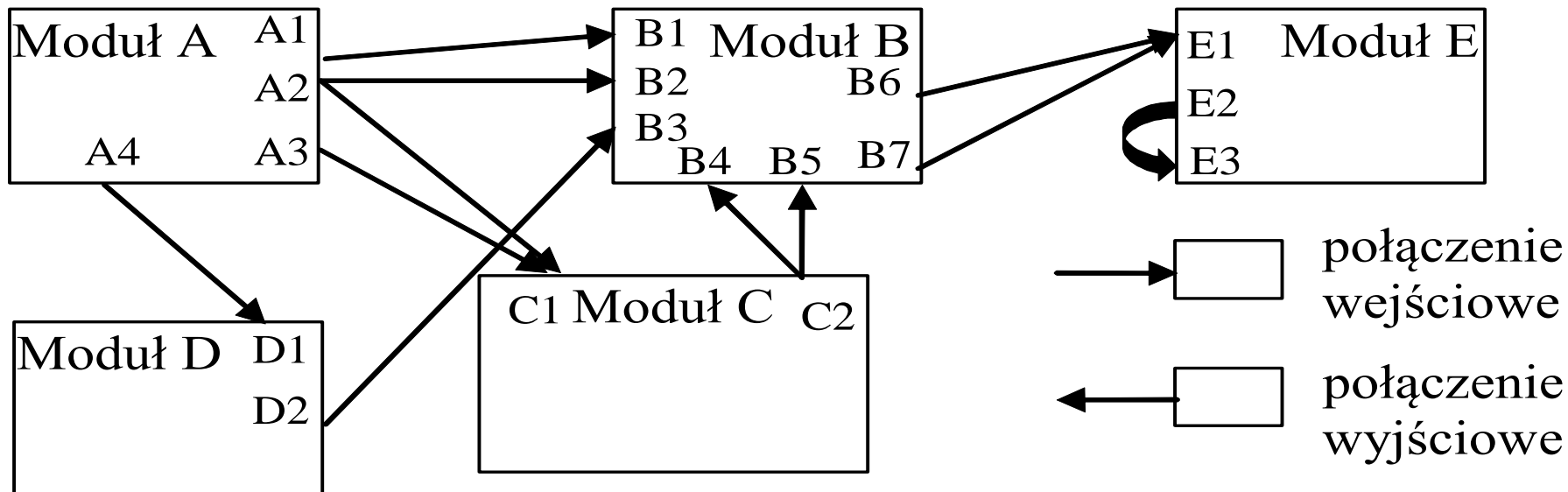
Ca - metryka połączeń wejściowych

Metryka CA wyznacza liczbę klas, które używają danej klasy przez wywołanie jej metod zwykłych lub wirtualnych (tyle razy liczonych, ile klas przesłania metodę), zastosowanie odwołania do zmiennej (wzajemne powiązanie między klasami jest liczone tylko raz) typu danej klasy i dziedziczonych przez nią atrybutów, przez argumenty metod typu danej klasy, wyniki typu danej klasy zwracane przez return oraz wyjątki– definicja powiązań wejściowych jest taka sama jak CBO.

Przykład rozwiązania dla modułu A (rysunek z następnego slajdu)

- Moduł A zawiera elementy łączące wyjściowe: A1, A2, A3, A4. Moduł B dla modułu A zawiera łączące elementy wejściowe B1, B2, moduł C zawiera łączący element wejściowy C1 oraz moduł D zawiera element wejściowy łączący D1 oraz:
 - A1 łączy się z B1
 - A2 łączy się B2, C1
 - A3 łączy się C1
 - A4 łączy się D1
- $RS = \{A1, A2, A3, A4\} \cup \{B1, B2\} \cup \{D1\} \cup \{C1\} = \{A1, A2, A3, A4, B1, B2, D1, C1\}$
- $RFC = |RS| = 8$
- $Fan-out = |\{<A1, B1>, <A2, C1>, <A4, D1>\}| = 3$ //dowolny element wejściowy
- $Fan-in = |\{\}| = 0$
- $R = \{<A1, B1>, <A2, B2>, <A2, C1>, <A3, C1>, <A4, D1>\}$
- $|R| = 5$

Przykłady metryk międzymodułowych dla modułów A, B, C, D, E cd.



	A	B	C	D	E
Fan-out	3	1	1	1	1
Fan-in	0	3	1	1	2
RFC	8	3	3	2	2
R	5	2	2	1	1

5. Metryki złożoności modułowej

Wzmocnienie powiązań wewnątrz-modułowych prowadzi do zmniejszenia oddziaływań między modułami oraz poprawy struktury oprogramowania.

Metryki rozmiaru

SLOC

- Jest to liczba wierszy kodu źródłowego programu liczona niezależnie od liczby instrukcji lub fragmentów instrukcji znajdujących się w każdym wierszu. Nie wlicza się wierszy z komentarzami lub pustych wierszy.
- SLOC jest powszechnie używaną metryką do szacowania nakładów pracy nad programem oraz jest mocno skorelowana z testowalnością, konserwowalnością i zrozumiałością.
- Zakres wartości 5 -1000 linii

S/C

- Metryka ta jest liczbą wszystkich elementów programu należących do bloków logicznych:
 - inicjowanie zmiennych sterujących
 - porównanie
 - zwiększanie zmiennej sterującej
 - liczba instrukcji w każdym bloku
- ```
int i=0
i <10
i++
for (;;) {...}
```

# Żetony

- Jest to zbiór metryk, które określają liczbę:
- $\eta_1$  - liczbę typów operatorów (słownik typów operatorów), czyli liczbę: operatorów predefiniowanych (logicznych, arytmetycznych, przypisania, relacyjnych itp.), słowa kluczowe instrukcji (**while**, **if**, **else**, **do**), nazwy funkcji
- $\eta_2$  - liczbę typów argumentów (słownik typów argumentów), czyli liczbę: wszystkich symboli reprezentujących dane przy deklaracji i definicji
- $\eta_3$  - liczbę wszystkich wystąpień operatorów
- $\eta_4$  - liczbę wszystkich wystąpień argumentów

## NPM - liczba metod publicznych

- Metryka wyznacza liczbę metod publicznych, która pozwala wyznaczyć miarę rozmiaru API pakietu, w którym znajduje się klasa.

# WMC - Liczba metod w klasie

## Zakres wartości (1 - 50)

- **Suma złożoności metod** w klasie (struktura logiczna i rozmiar)

$$WMC = \sum_{i=1}^n c_i$$

- gdzie  $c_i$  jest statyczną złożonością każdej z  $i$  - metod (złożoność cyklomatyczna materiał podany dalej). Jeżeli  $c_i$  jest równe 1, wtedy  $WMC$  jest równe liczbie metod  $n$ .  $WMC$  maleje przy wykorzystaniu polimorfizmu i dziedziczenia

### Uwagi:

- Zbyt duża wartość metryki powoduje w klasie więcej błędów
- Zbyt duża wartość oznacza mniejszą wieloużywalność klasy
- Zbyt duża wartość powoduje mniejsze zrozumienie odpowiedzialności klasy

# DIT - Głębokość dziedziczenia

## Zakres wartości (0 - 5)

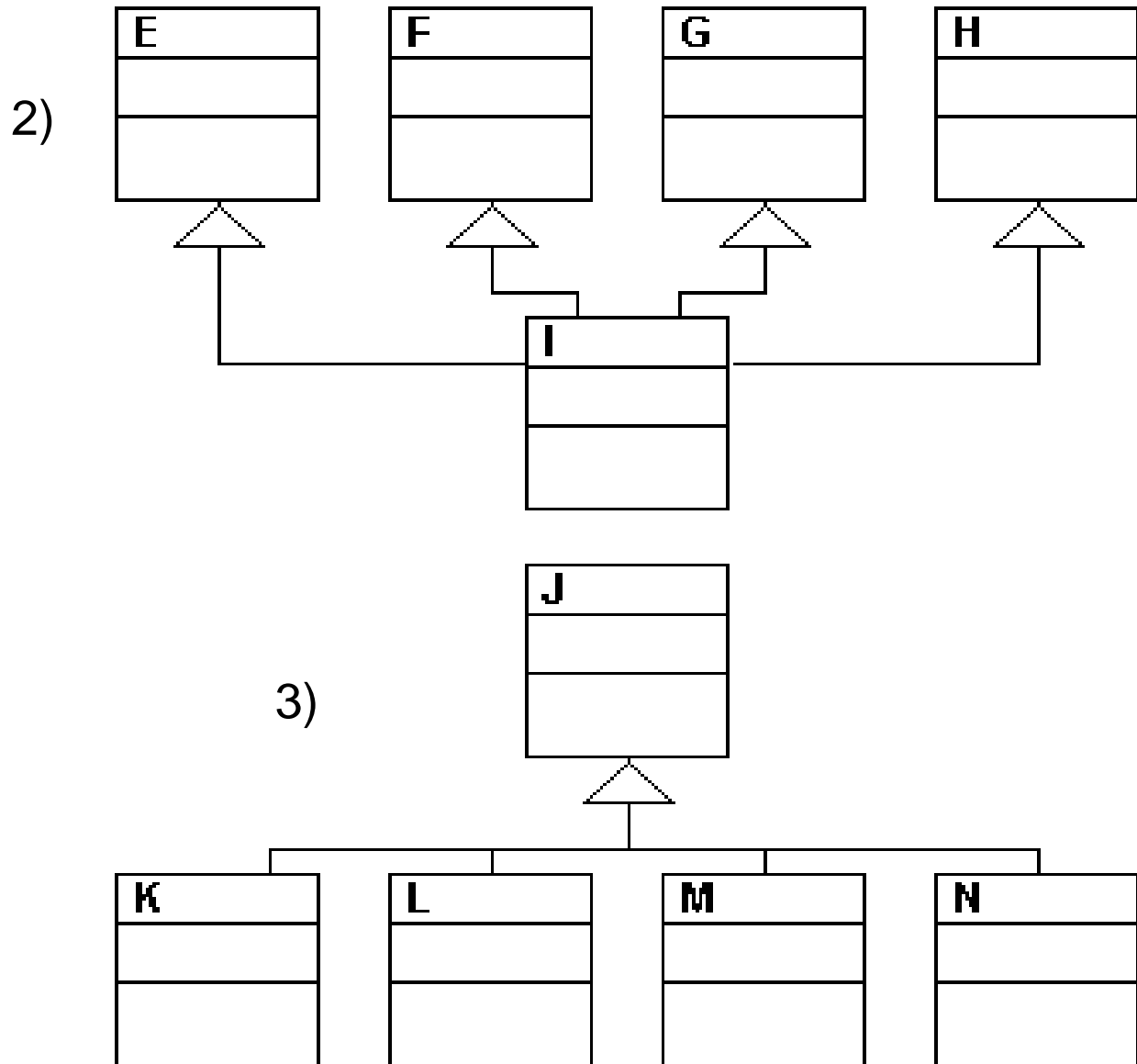
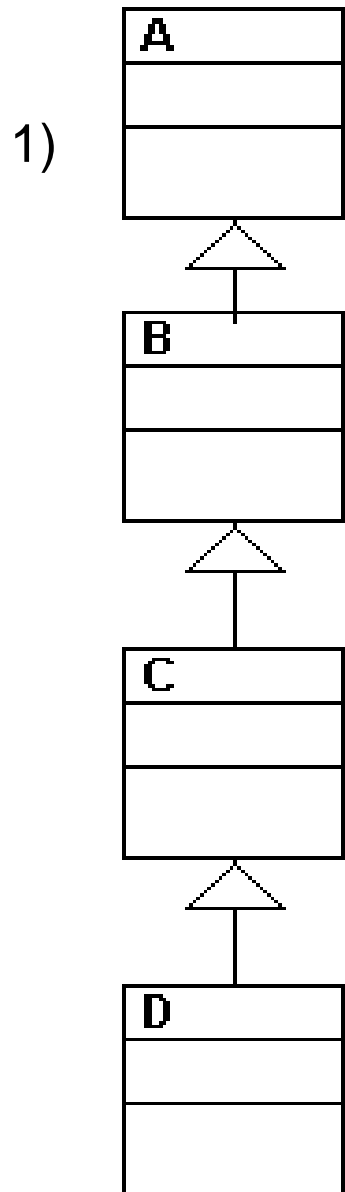
- czyli liczba poziomów w drzewie dziedziczenia odniesiona do liczby klas, określająca zakres dziedziczenia (rozmiar)

$$DIT = \frac{\sum \text{glebokosc dziedziczenia}}{\text{calkowita liczba klas}}$$

*Uwagi:*

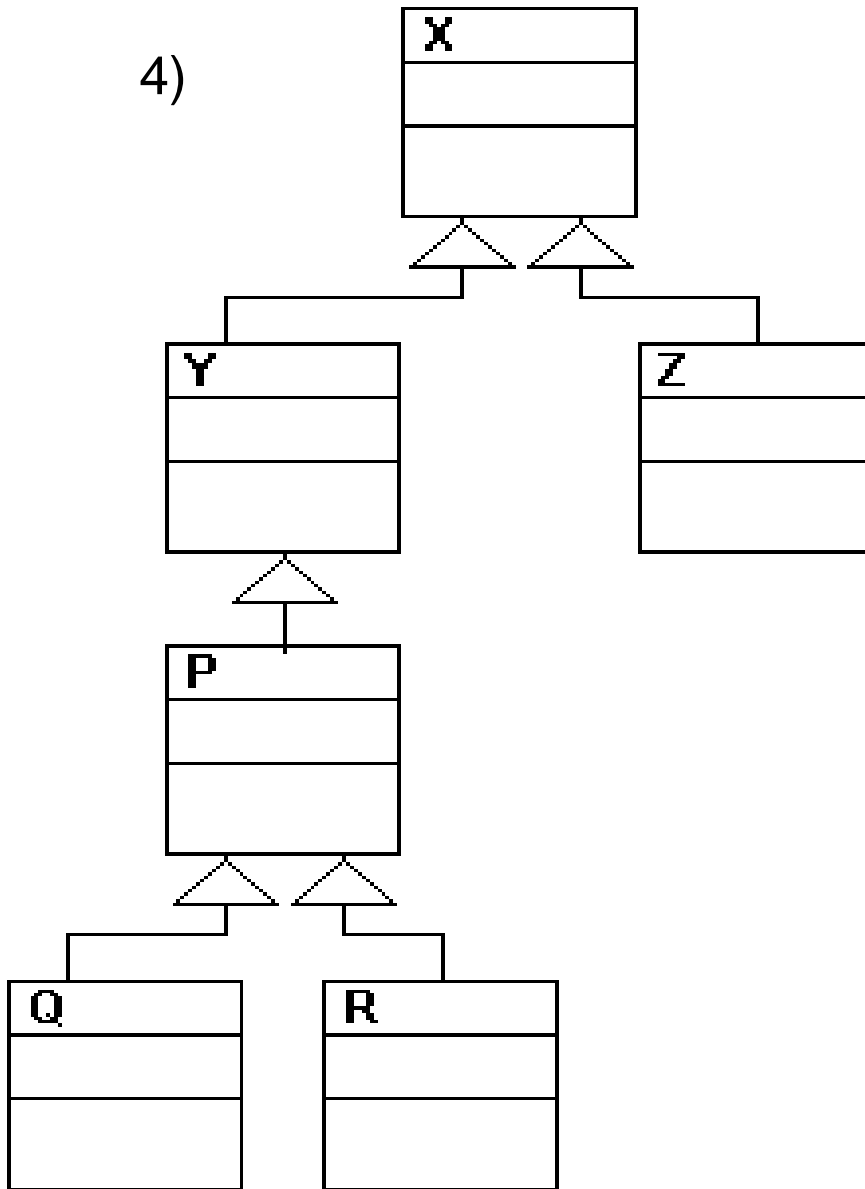
1. Przy głębokim drzewie dziedziczenia rośnie wieloużywalność
2. Przy głębokim drzewie dziedziczenia rośnie też liczba błędów, szczególnie w klasach należących do środkowych poziomów dziedziczenia

# (1) Przykłady modeli do pomiaru metryk dziedziczenia

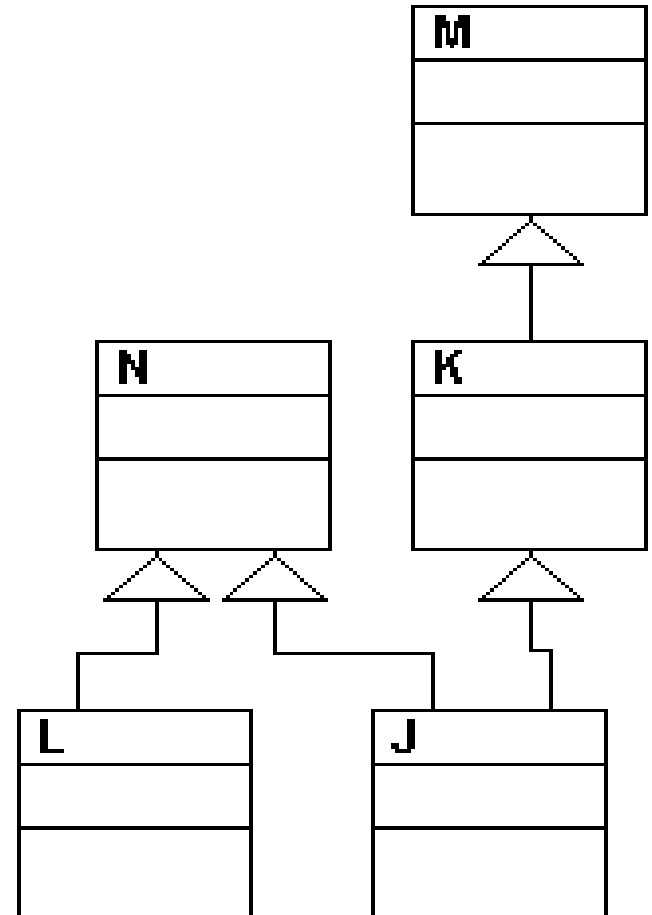


## (2) Przykłady modeli do pomiaru metryk dziedziczenia

4)



5)



| Metryka  | S (specialization) | U (reuse) | DIT                         |
|----------|--------------------|-----------|-----------------------------|
| Przykład |                    |           |                             |
| 1*       | 1/3 → 0            | 3/4 → 1   | (0+1+2+3)/4=1.5             |
| 2*       | 1/4 → 0            | 4/5 → 1   | (0+0+0+0+(1+1+1+1)/4)/5=0.2 |
| 3        | 4/1 → μ            | 1/5 → 0   | (0+1+1+1+1)/5=0.8           |
| 4        | 3/3                | 3/6       | (0+1+1+2+3+3)/6=1.5         |
| 5        | 2/3                | 3/5       | (0+0+1+(1+2)/2+1)/5=0.7     |

$$DIT = \frac{\sum \text{glebokosc dziedziczenia}}{\text{całkowita liczba klas}}$$

$$S = \frac{\text{liczba PodTypów}}{\text{liczba NadTypów}}$$

$$U = \frac{\text{liczba NadTypów}}{\text{całkowita liczba klas}}$$

Przykłady 1 i 2 reprezentują ubogi schemat dziedziczenia.

Wartości  $U$  bliska 1 oraz  $S$  bliską 0 określają liniowy model dziedziczenia. Wartości  $U \ll 1$  oraz  $S \gg 1$  oznaczają pożądaną wartość.

# NOC – liczba klas dziedziczących

## Zakres wartości (0..10)

### Uwagi

1. Zbyt dużo podklas oznacza dużo testowania
2. Zbyt dużo podklas może powodować błędne użycie tych podklas



# Metryki logicznej struktury programu, czyli przepływu sterowania

## Liczby cyklomatyczne McCabe

### Zakres wartości (1 -10)

$$VLI(G) = e - n + p + 1$$

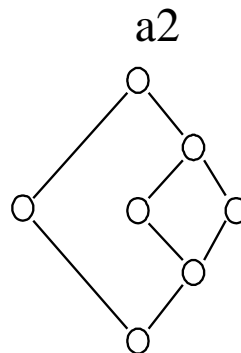
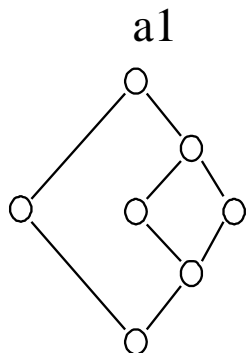
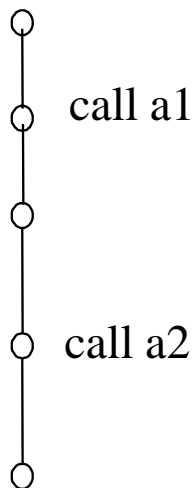
Liczba ta jest wyznaczana na podstawie grafu przedstawiającego drogi sterowania w programie, gdzie  $n$  jest liczbą wierzchołków grafu reprezentujących poszczególne instrukcje, w tym wywołania funkcji,  $e$  jest liczbą krawędzi grafu reprezentujących połączenia poszczególnych realizacji instrukcji,  $p$  jest liczbą podgrafów rozłącznych, a każda funkcja stanowi niezależny podgraf, którego wywołanie jako wierzchołek jest umieszczony w innym podgrafie.

$$V(G) = e - n + 2 * p$$

Metryka  $V(G)$  uwypukla istnienie funkcji za pomocą składnika  $2 * p$ ,  $VLI(G)$  natomiast wywołanie funkcji traktuje na równi z innymi instrukcjami.

# (1) Przykład prezentujący obliczenia metryk MC Cabe

a)



**Cała aplikacja**

**a)**  $e=20, n=19, p=3$

$$V(G) = e - n + 2 * p = 20 - 19 + 2 * 3 = 7$$

$$V_{LI}(G) = e - n + p + 1 = 20 - 19 + 3 + 1 = 5$$

**b)**  $e=23, n=20, p=1$

$$V(G) = V_{LI}(G)$$

$$= e - n + 2 = e - n + 2 * p$$

$$= 23 - 20 + 2 = 5$$

**Metody a1 i a2 (przypadki a i b):**

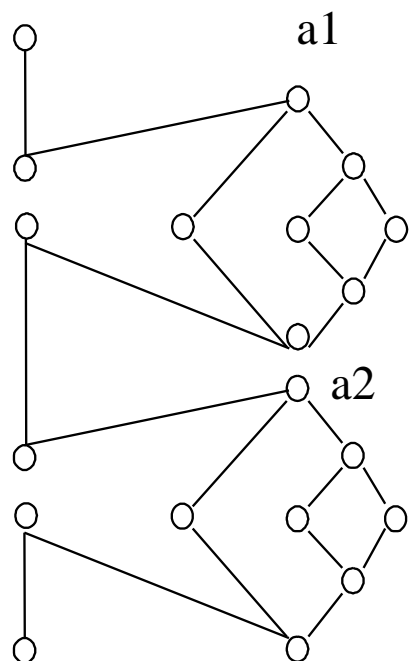
$e=8, n=7, p=1$

$$V(G) = V_{LI}(G) =$$

$$= e - n + 2 = e - n + 2 * p$$

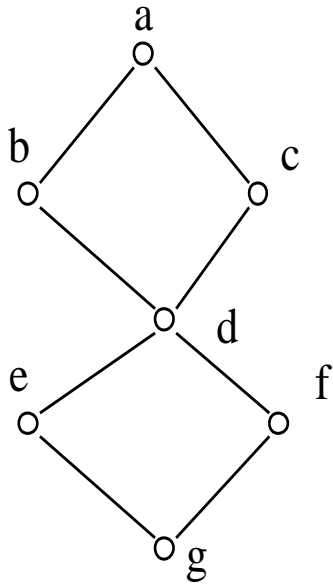
$$= 8 - 7 + 2 = 3$$

b)



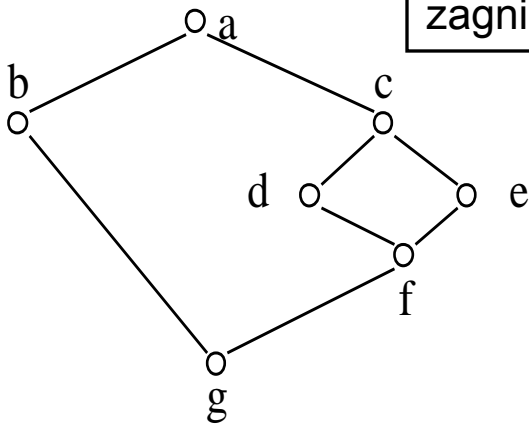
|                     | Metoda a1 | Metoda a2 | Całość          |
|---------------------|-----------|-----------|-----------------|
| a                   |           |           |                 |
| V(G)                | 3         | 3         | 7               |
| V <sub>LI</sub> (G) | 3         | 3         | 5               |
| b                   |           |           |                 |
| V(G)                | 3         | 3         | 5 <sub>26</sub> |
| V <sub>LI</sub> (G) | 3         | 3         | 5               |

## (2) Przykład prezentujący obliczenia metryk MC Cabe



Zgodnie z aksjomatem 7, pętla zagnieżdżona powinna mieć złożoność różną od programu z dwiema sekwencyjnie wykonywanymi pętlami.

Jednak zarówno SLOC,  $V(G)$ ,  $VLI(G)$  są identyczne w obu rozwiązaniach, natomiast różne są wartości metryki S/C. Wg metryki S/C bardziej złożony jest program z zagnieżdżoną pętlą.



dwie pętle sekwencyjne

a: **while** ( $x \geq 0$ )

c: {  $x=x-y$ ; } (gdy  $a==true$ )

b: (gdy  $a==false$ )

d: **while** ( $y \geq 10$ ) (koniec a)

f: {  $x=x+1$ ; } (gdy  $d==true$ )

$y=y-1$ ;

}

e: (gdy  $d==false$ )

g: (koniec d, koniec programu)

**$V(G)=e-n+2*p=3$**

**$VLI(G)=e-n+p+1=8-7+2=3$**

**SLOC=7**

**S/C=7**

b) podwójna pętla zagnieżdżona

a: **while** ( $x \geq 0$ )

{  $x=x-y$ ; } (gdy  $a==true$ )

c: **while** ( $y \geq 10$ ) (gdy  $a==true$ )

e: {  $x=x+1$ ; } (gdy  $c==true$  i  $a==true$ )

$y=y-1$ ;

d: (gdy  $c==false$  i  $a==true$ )

f: (koniec c i  $a==true$ )

}

b: (gdy  $a==false$ )

g: (koniec a, koniec programu)

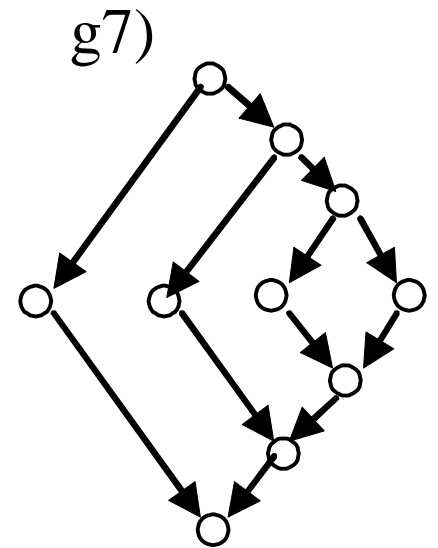
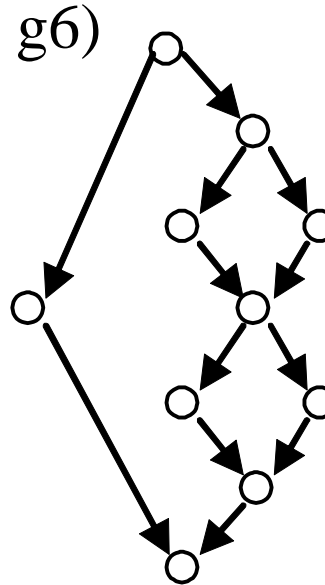
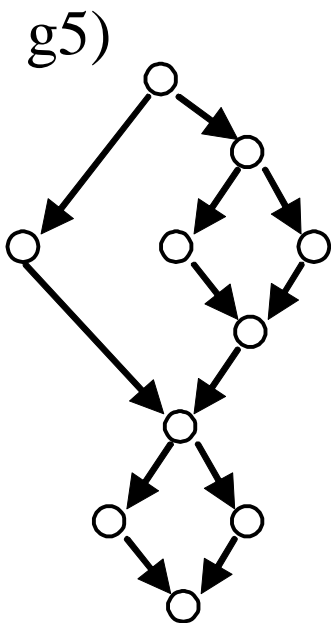
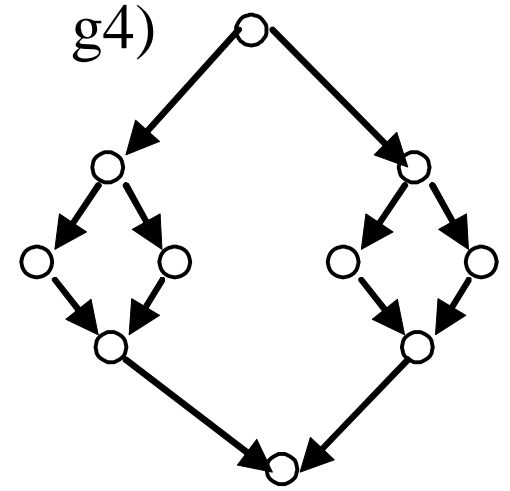
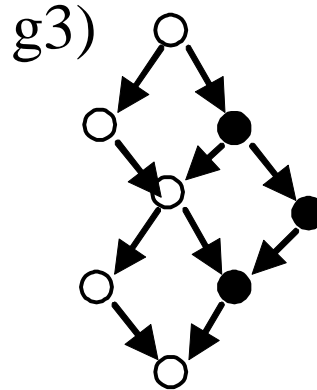
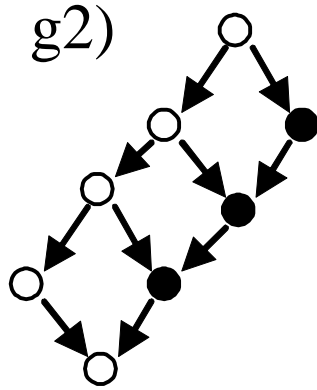
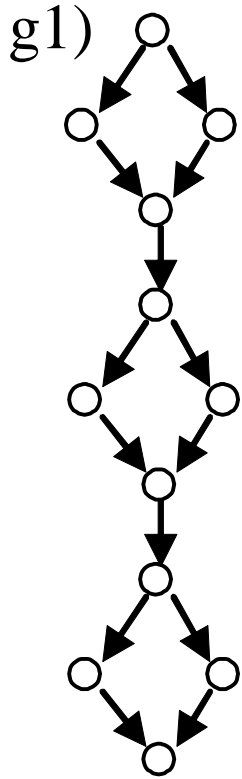
**$V(G)=e-n+2*p=3$**

**$VLI(G)=e-n+p+1=8-7+2=3$**

**SLOC=7**

**S/C=9**

(3) Dodatek\* *Metryki P1 , P2 (Henderson-Sellers)*



#### (4) Dodatek\* *Metryki złożoności cykloamatycznej P1 , P2 (Henderson-Sellers)*

$$P1 = a1 * V(G) + a2 * N^* + a3 * (ve - 1) \quad \text{- wersja addytywna}$$

$$P2 = [a1 * V(G) + a2 * N^*] * a3 * ve \quad \text{- wersja multiplikatywna}$$

- gdzie:  $a1$  ,  $a2$  ,  $a3$  są współczynnikami, które dla uproszczenia mają wartość 1
- $N^* = \sum_i P(i)$ , gdzie  $p(i)$  jest długością ścieżki zagnieźdzenia i-tego predykatu w instrukcjach decyzyjnych
- $ve$  – liczba cykloamatyczna wyznaczana dla programów niestukturalnych
- $ve = d+1$ , gdzie  $d$  jest liczbą niestukturalnych rozgałęzień w programie (•)

|       | g1 | <b>g2</b> | <b>g3</b> | g4 | g5 | g6 | g7 |
|-------|----|-----------|-----------|----|----|----|----|
| V(G)  | 4  | <b>4</b>  | <b>4</b>  | 4  | 4  | 4  | 4  |
| $v_e$ | 1  | <b>4</b>  | <b>4</b>  | 1  | 1  | 1  | 1  |
| P1    | 4  | <b>7</b>  | <b>7</b>  | 6  | 5  | 6  | 7  |
| P2    | 4  | <b>16</b> | <b>16</b> | 6  | 5  | 6  | 7  |

# Metryki spójności klasy

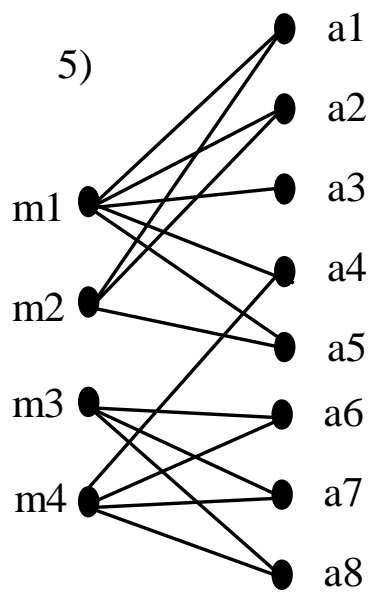
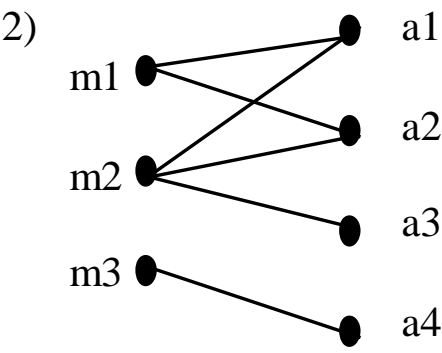
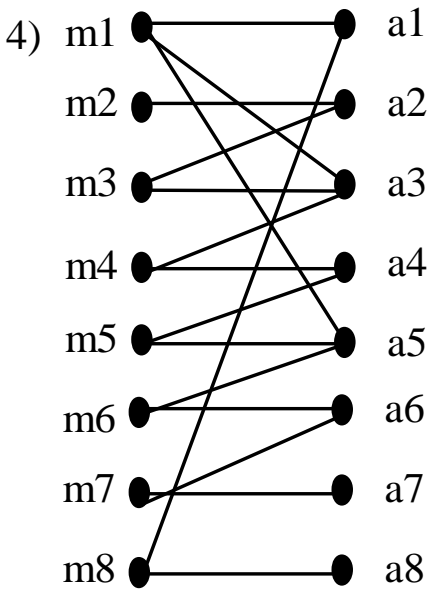
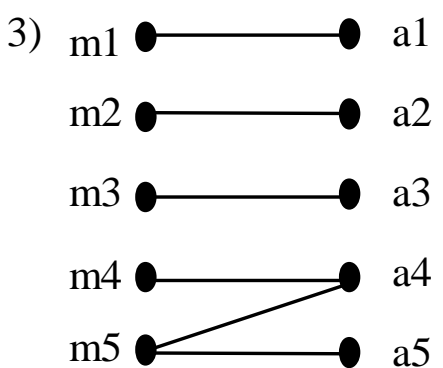
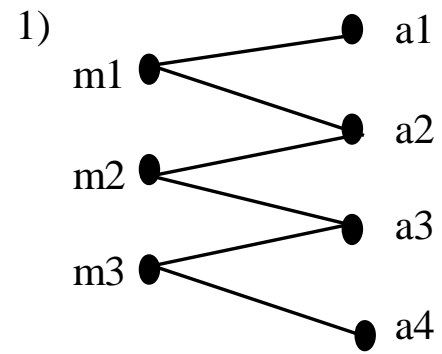
**LCOM1** – metryka wyznacza sumę  $P$  zbioru wszystkich par metod operujących na zbiorach rozłącznych atrybutów oraz sumę  $Q$  zbioru wszystkich par metod operujących na zbiorach spójnych atrybutów. Różnica mocy tych zbiorów jest wartością metryki, gdy moc  $|P|$  jest większa od mocy  $|Q|$ , w przeciwnym wypadku jest równa 0.

Jeśli klasa jest minimalnie spójna (żadna metoda nie jest powiązana z inną metodą i liczba metod jest równa  $n$ ). Wtedy  $|P| = (n-1)*n/2$  i  $|Q|=0$ , czyli  $LCOM1=(n-1)*n/2$

Uwagi:

- 1) Duża wartość metryki oznacza trudność testowania,
- 2) jednak mała wartość lub równa 0 nie zawsze oznacza klasę poprawnie zbudowaną.
- 3) Zbyt wiele różnych klas ma tę samą wartość metryki.
- 4) Brak modelowania property i uwzględnienia wywoływania metody przez metodę

# Grafy dwudzielne jako modele klas do wyznaczania metryki LCOM



# (1) Przykłady obliczeń metryki LCOM1

## 1) – trzy metody

- Metoda 1 ma zbiór atrybutów  $I_1 = \{a_1, a_2\}$
- Metoda 2 ma zbiór atrybutów  $I_2 = \{a_2, a_3\}$
- Metoda 3 ma zbiór atrybutów:  $I_3 = \{a_3, a_4\}$
- Zbiór rozłącznych par:  $P = \{(I_1, I_3)\} \rightarrow |P| = 1$
- Zbiór spójnych par:  $Q = \{(I_1, I_2), (I_2, I_3)\} \rightarrow |Q| = 2$
- **LCOM = 0 dla  $|P| \leq |Q|$**

## 2) - trzy metody

- Metoda 1 ma zbiór atrybutów  $I_1 = \{a_1, a_2\}$
- Metoda 2 ma zbiór atrybutów  $I_2 = \{a_1, a_2, a_3\}$
- Metoda 3 ma zbiór atrybutów:  $I_3 = \{a_4\}$
- Zbiór rozłącznych par:  $P = \{(I_1, I_3), (I_2, I_3)\} \rightarrow |P| = 2$
- Zbiór spójnych par:  $Q = \{(I_1, I_2)\} \rightarrow |Q| = 1$
- **LCOM =  $|P| - |Q| = 2 - 1 = 1$  dla  $|P| > |Q|$**

## 3) – pięć metod

- Metoda 1 ma zbiór atrybutów  $I_1 = \{a_1\}$
- Metoda 2 ma zbiór atrybutów  $I_2 = \{a_2\}$
- Metoda 3 ma zbiór atrybutów:  $I_3 = \{a_3\}$
- Metoda 4 ma zbiór atrybutów:  $I_4 = \{a_4\}$
- Metoda 5 ma zbiór atrybutów:  $I_5 = \{a_4, a_5\}$
- Zbiór rozłącznych par:  $P = \{(I_1, I_2), (I_1, I_3), (I_1, I_4), (I_1, I_5), (I_2, I_3), (I_2, I_4), (I_2, I_5), (I_3, I_4), (I_3, I_5)\} \rightarrow |P| = 9$
- Zbiór spójnych par:  $Q = \{(I_4, I_5)\} \rightarrow |Q| = 1$
- **LCOM =  $|P| - |Q| = 9 - 1 = 8$  dla  $|P| > |Q|$**



## (2) Przykłady obliczeń metryki LCOM1

### 5) – cztery metody

- Metoda 1 ma zbiór atrybutów  $I_1 = \{a_1, a_2, a_3, a_4, a_5\}$
- Metoda 2 ma zbiór atrybutów  $I_2 = \{a_1, a_2, a_5\}$
- Metoda 3 ma zbiór atrybutów:  $I_3 = \{a_6, a_7, a_8\}$
- Metoda 4 ma zbiór atrybutów:  $I_4 = \{a_4, a_6, a_7, a_8\}$
- Zbiór rozłącznych par:  $P = \{(I_1, I_3), (I_2, I_3), (I_2, I_4)\} \rightarrow |P| = 3$
- Zbiór spójnych par:  $Q = \{(I_1, I_2), (I_1, I_4), (I_3, I_4)\} \rightarrow |Q| = 3$
- **LCOM = 0 dla  $|P| \leq |Q|$**

### 4) – osiem metod

- Metoda 1 ma zbiór atrybutów  $I_1 = \{a_1, a_3, a_5\}$
- Metoda 2 ma zbiór atrybutów  $I_2 = \{a_2\}$
- Metoda 3 ma zbiór atrybutów:  $I_3 = \{a_2, a_3\}$
- Metoda 4 ma zbiór atrybutów:  $I_4 = \{a_3, a_4\}$
- Metoda 5 ma zbiór atrybutów:  $I_5 = \{a_4, a_5\}$
- Metoda 6 ma zbiór atrybutów:  $I_6 = \{a_5, a_6\}$
- Metoda 7 ma zbiór atrybutów:  $I_7 = \{a_6, a_7\}$
- Metoda 8 ma zbiór atrybutów:  $I_8 = \{a_1, a_8\}$
- Zbiór rozłącznych par:  $P = \{(I_1, I_2), (I_1, I_7), (I_2, I_4), (I_2, I_5), (I_2, I_6), (I_2, I_7), (I_2, I_8), (I_3, I_5), (I_3, I_6), (I_3, I_7), (I_3, I_8), (I_4, I_6), (I_4, I_7), (I_4, I_8), (I_5, I_7), (I_5, I_8), (I_6, I_8), (I_7, I_8)\}$   
 $\rightarrow |P| = 18$
- Zbiór spójnych par:  $Q = \{(I_1, I_3), (I_1, I_4), (I_1, I_5), (I_1, I_6), (I_1, I_8), (I_2, I_3), (I_3, I_4), (I_4, I_5), (I_5, I_6), (I_6, I_7),\}$   
 $\rightarrow |Q| = 10$
- **LCOM =  $|P| - |Q| = 18 - 10 = 8$  dla  $|P| > |Q|$**

# Rozszerzenie definicji metryk spójności LCOM (1)

## Metryka LCOM2 (Constantine & Graham, Henderson-Sellers)

$$LCOM2 = 1 - \frac{\left(\frac{1}{a} \sum_{j=1}^a \mu(A_j)\right)}{m} = 1 - \frac{r}{m * a}$$

- gdzie  $m$  jest liczbą wierzchołków zbioru  $M$  metod,  $a$  jest liczbą wierzchołków  $A$  atrybutów, natomiast wyrażenie  $\mu(A_j)$  liczbą krawędzi grafu wiążącą atrybut  $A_j$  z określoną liczbą metod (elementy zbioru  $R$ ).
- Maksymalna i zarazem najlepsza wartość spójności LCOM2 oznacza wartość 0 metryk, co uzyskuje się przy grafie pełnym ( $r = |M| * |A|$  krawędzi).
- Wartość metryki LCOM2 zawarta między „0..mniejszy od 1” oznacza obiektowy model klasy, jednak warta bliska 1 oznacza najgorszy przypadek klasy.
- W metryce LCOM2 muszą przynajmniej istnieć jedna metoda i jeden atrybut.

| Lp       | $m$      | $a$      | $r$       | $k$      | LCOM1    | LCOM2         | LCOM3         |
|----------|----------|----------|-----------|----------|----------|---------------|---------------|
| 1        | 3        | 4        | 6         | 1        | 0        | 0.5           | 0.75          |
| 2        | 3        | 4        | 6         | 2        | 1        | 0.5           | 0.75          |
| 3        | 5        | 5        | 6         | 4        | 8        | 0.76          | 0.95          |
| 4        | 8        | 8        | 16        | 1        | 8        | 0.75          | 0.8571        |
| <b>5</b> | <b>4</b> | <b>8</b> | <b>15</b> | <b>1</b> | <b>0</b> | <b>0.5313</b> | <b>0.7083</b> |

## Rozszerzenie definicji metryk spójności LCOM (2)

### Metryka LCOM3 (Constantine & Graham, Henderson-Sellers)

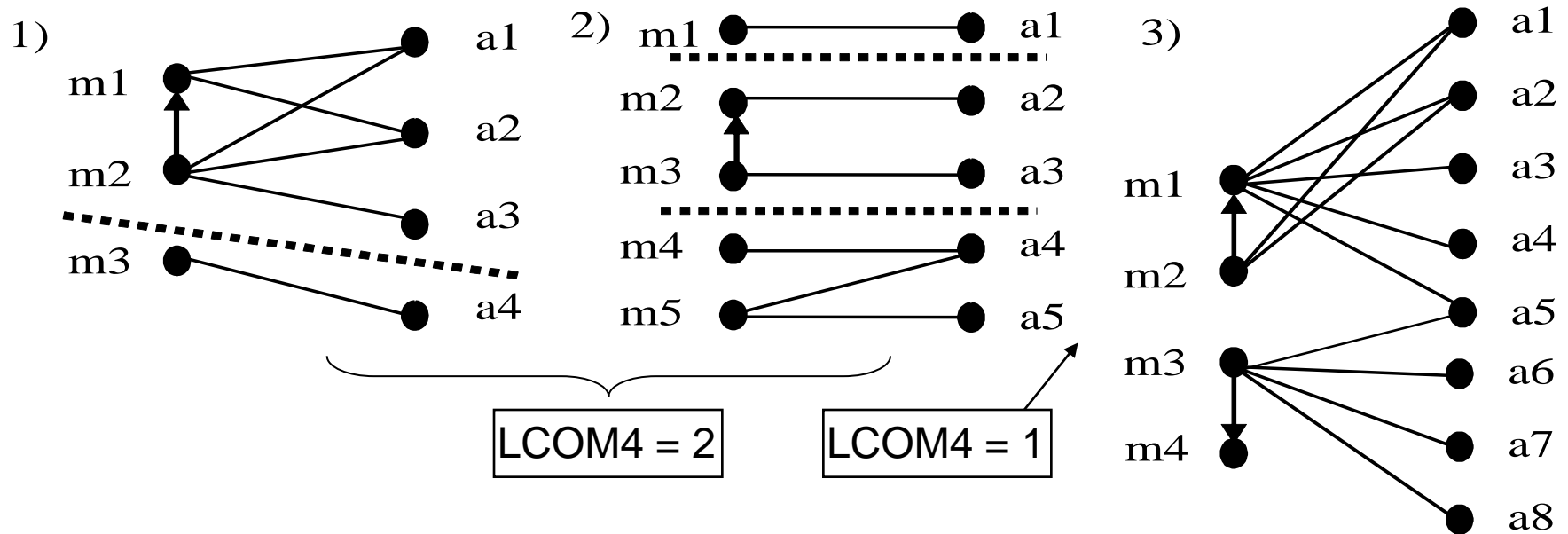
Zakres wartości „0..0.2”.

$$LCOM3 = \frac{\left(\frac{1}{a} \sum_{j=1}^a \mu(A_j)\right) - m}{1 - m} = \frac{r - m}{1 - m}$$

- gdzie  $m$  jest liczbą wierzchołków zbioru  $M$  metod,  $a$  jest liczbą wierzchołków  $A$  atrybutów, natomiast wyrażenie  $\mu(A_j)$  liczbą krawędzi grafu wiążącą atrybut  $A_j$  z określoną liczbą metod (elementy zbioru  $R$ ).
- Maksymalna i zarazem najlepsza wartość spójności LCOM3 oznacza wartość 0 metryk, co uzyskuje się przy grafie pełnym ( $r=|M|^*|A|$  krawędzi).
- Wartość metryki LCOM3 zawarta między „0..1” oznacza obiektowy model klasy (wartość 1 oznacza minimalnie spójną klasę – równa liczby metod i atrybutów). Dopuszczalny zakres „0..0.2”.
- W metryce LCOM3 w klasie nie może istnieć tylko jedna metoda i musi być przynajmniej jeden atrybut.

# Rozszerzenie definicji metryk spójności LCOM (3)

## LCOM4 - (Hitz & Montazeri)



- **LCOM4** mierzy liczbę „połączonych komponentów” w klasie. „Połączony komponent” jest zbiorem połączonych metod (zbiór takich metod  $a$  i  $b$ , gdzie metoda  $a$  wywołuje metodę  $b$  lub metoda  $b$  wywołuje metodę  $a$ , lub obie metody  $a$  i  $b$  wywołują ten sam atrybut klasy) i atrybutów, przy czym dopuszcza się jeden taki komponent klasy.
- Jeśli wartość metryki jest równa 2 lub więcej, należy klasę podzielić na dwie klasy lub więcej klas, tak aby posiadała tylko jeden „połączony komponent”.

## Przykład metryk CK trzech systemów

| <b>System analyzed</b> | <b>Java</b> | <b>Java</b> | <b>C++</b>          |
|------------------------|-------------|-------------|---------------------|
| <b>Classes</b>         | 46          | 1000        | 1617                |
| <b>Lines</b>           | 50,000      | 300,000     | 500,000             |
| <b>Quality</b>         | "Low"       | "High"      | "Medium"            |
| <b>CBO</b>             | 2.48        | 1.25        | 2.09                |
| <b>LCOM1</b>           | 447.65      | 78.34       | 113.94              |
| <b>RFC</b>             | 80.39       | 43.84       | 28.60               |
| <b>NOC</b>             | 0.07        | 0.35        | 0.39                |
| <b>DIT</b>             | 0.37        | 0.97        | 1.02                |
| <b>WMC</b>             | 45.7        | 11.10       | 23,97 <sub>37</sub> |