Złożoność strukturalna programów, pomiary oprogramowania - metryki

7.1. Podstawowe definicje

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 osiąganiu 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).

co opowiada następującym *atrybutom zewnętrznym oprogramowania*, wynikających *ze złożoności psychologicznej:*

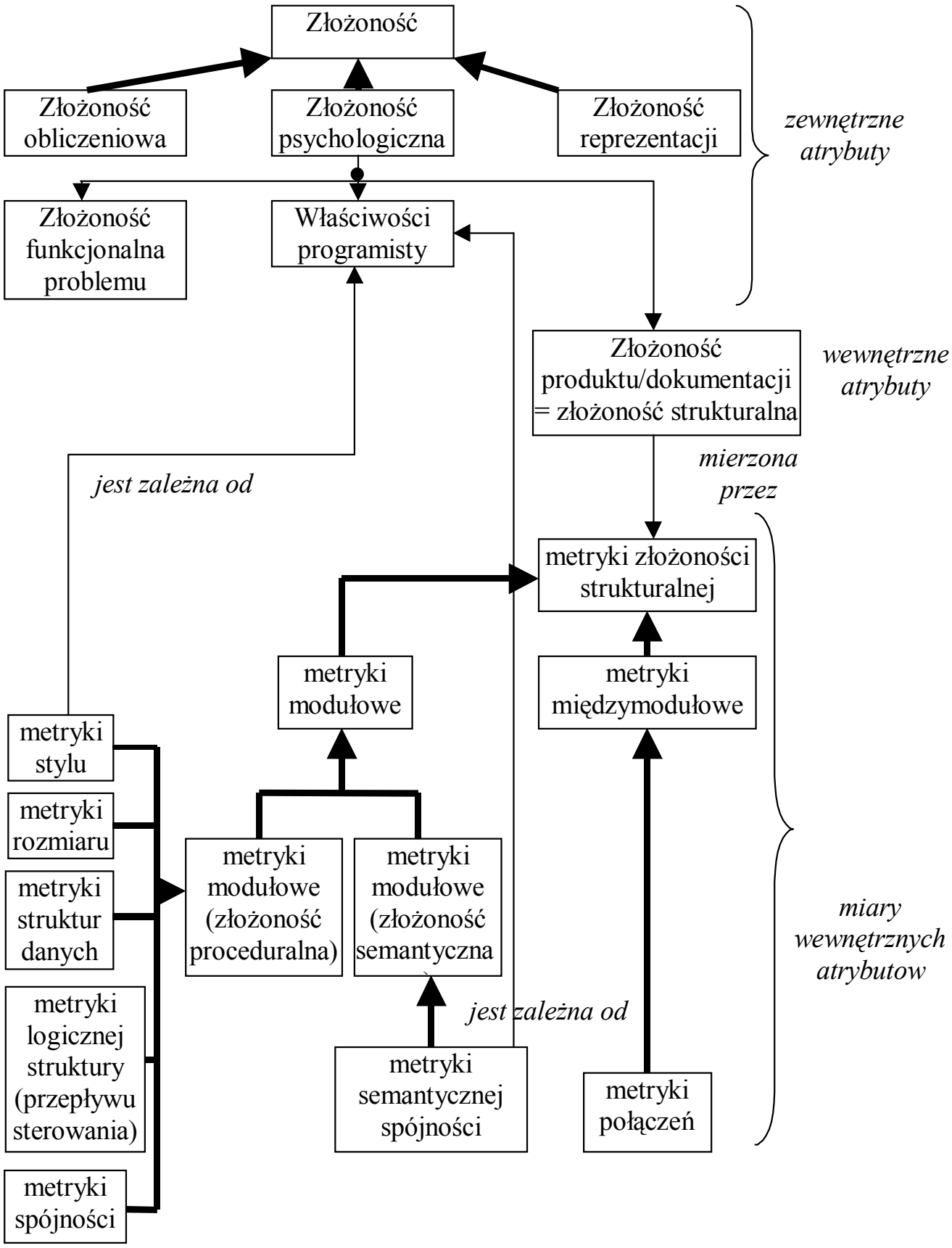
- testowalności, a więc również niezawodności,
- stopnia osiągniętej abstrakcji
- zrozumiałości programu
- stopnia pielęgnacji
- wielożywalności.

Atrybuty zewnętrzne zależą od *atrybutów wewnętrznych oprogramowania* i są wyrażane w postaci obiektywnych miar tych atrybutów.

Wyróżnia się następujące *atrybuty wewnętrzne* oprogramowania:

- *charakterystyki międzymodułowe* czyli wszelkie związki między modułami (przekazywanie sterowania i parametrów, wspólne korzystanie z pól danych, przy projektowaniu jednego modułu uwzględnia się właściwości innego modułu
- *charakterystyki 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.

Te *atrybuty wewnętrzne* są wyrażane za pomocą 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.



Rys. 1. Klasyfikacja pojęć złożoności oprogramowania i podanie związku między zewnętrznymi atrybutami oprogramowania i miarami wewnętrznych atrybutów [12]

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 ≡ Q tzn. spełniające te same funkcje, lecz o różnej złożoności.
5. Złożoność programów P i Q połączonych jest niewiększa 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.

Podstawowe metryki

Poszczególne metryki złożoności strukturalnej mogą być wyznaczane w dowolnym stadium rozwoju oprogramowania, jednak to rzutuje na wybór elementów produktu i rodzaj wyrażenia. Opis dotyczy metryk operujących na elementach programów napisanych w języku C++.

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