

Dr Katarzyna Grzesiak-Kopeć

Inżynieria oprogramowania



8. Miary oprogramowania



Plan wykładu

- Tworzenie oprogramowania
- Najlepsze praktyki IO
- Inżynieria wymagań
- Technologia obiektowa i język UML
- Techniki IO
- Metodyki zwinne
- Refaktoryzacja
- **Mierzenie oprogramowania**
- Jakość oprogramowania
- Programowanie strukturalne
- Modelowanie analityczne
- Wprowadzenie do testowania

O co chodzi?

- Pomiar, miara, metryka
- Co i po co mierzemy
- Metryki produktu
 - Rozmiar
 - Funkcjonalność
 - Złożoność



Pomiar, miara, metryka

- Pomiar

- Czynność, dzięki której można poznać wielkość

- Miara

- Ilościowe wskazanie czy i w jakim stopniu dany produkt lub proces wykazuje analizowaną cechę
 - Liczba, wymiar, pojemność, rozmiar cechy
 - Liczba błędów



Pomiar, miara, metryka

- Metryka

- Ilościowa miara stopnia w jakim system, komponent lub proces posiada dany atrybut
- Liczba błędów znaleziona przez osobę w ciągu godziny



Po co mierzyć?

- Określenie jakości produktu lub procesu
- Przewidzenie jakości produktu lub procesu
- Poprawa jakości produktu lub procesu

You can neither predict nor control what you cannot measure.

Tom DeMarco

Po co mierzyć?

- Kontrola
 - Zrozumienie
 - Porównanie
 - Predykcja
-

UDOSKONALANIE



Dobra metryka

- Prosta i łatwa do obliczenia
- Przekonująca empirycznie i intuicyjnie
- Spójna i obiektywna
- Zgodna z rzeczywistymi jednostkami i wymiarami
- Niezależna od języka programowania
- Pomocna w dokonaniu oceny jakości

L. Ejigou

Zalety stosowania metryk

- Znane kryteria sukcesu i porażki **ROZSĄDNE!**
 - produktu, procesu lub działalności ludzkiej
- Można ocenić efektywność wprowadzonych zmian
 - produktu, procesu lub działalności ludzkiej
- Ułatwione podejmowanie decyzji technicznych oraz kierowniczych
- Rozpoznawanie trendów i dokonywanie estymacji

M. Trzaska, Metryki w obiektowej io

Co mierzymy?

- Produkty
 - Rezultaty powstałe w czasie tworzenia oprogramowania
 - Kod źródłowy, specyfikacja, plan testów, dokumentacja...
- Procesy
 - Czynności wykonywane w trakcie tworzenia oprogramowania
 - Projektowanie, kodowanie, ...
- Zasoby
 - Zasoby wykorzystywane w trakcie tworzenia oprogramowania
 - Sprzęt, wiedza, metody, ludzie, ...

Jakie atrybuty mierzemy?

- Wewnętrzne
 - Mierzone elementy
 - Rozmiar, czas, cena, ...
- Zewnętrzne
 - Mierzone w stosunku do środowiska
 - Niezawodność, wydajność, skalowalność...

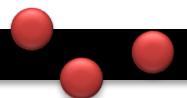


Przykład

| Element | Atrybut |
|-------------------------|---|
| Projekt | Liczba błędów wykrytych w trakcie inspekcji |
| Specyfikacja projektowa | Liczba stron |
| Kod źródłowy | Liczba linii kodu, liczba operacji |
| Zespół deweloperów | Rozmiar zespołu, przeciętne doświadczenie |

Rodzaje metryk

- Miary bezpośrednie
 - Koszt i liczba pracowników w projekcie
 - Liczba wierszy kodu źródłowego (LOC)
- Miary pośrednie
 - Defect density = liczba błędów / rozmiar
- Prognozowanie
 - Prognoza wielkości systemu na podstawie pomiaru funkcjonalności – punkty funkcyjne



Rodzaje metryk

- Nominalne
 - 3GL, 4GL
- Porządkowe
 - Wydajność: słaba, przeciętna, wysoka
- Przedziałowe
 - 1-150 LOC/day
- Względne
 - Produkt 2x większy od poprzedniego
- Bezwzględne
 - 500 000 LOC



Produkt vs proces

- Metryki procesu
 - Analiza paradygmatów, zadań, punktów milowych
 - Prowadzi do udoskonalania procesu – udoskonalanie długoterminowe
- Metryki produktu
 - Ocena stanu projektu
 - Uchwycenie potencjalnego ryzyka
 - Identyfikacja problemów
 - Dostosowanie "workflowu" oraz/lub zadań
 - Ocena umiejętności zespołu by zapewnić odpowiednią jakość produktu

Metryki produktu

- Rozmiar
- Funkcjonalność
- Złożoność



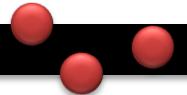
Pomiary rozmiaru produktu

- Linia/wiersz kodu, Conte 1986

Linią kodu nazywamy każdą linię tekstu napisanego programu, która nie jest ani komentarzem ani linią pustą, bez względu na liczbę instrukcji w linii. W szczególności są to linie zawierające nagłówki, deklaracje, wykonywalne i niewykonywalne instrukcje programu.

Rozmiar

- Rozmiar kodu źródłowego
- LOC, 1000 Lines Of Code (KLOC)
- Liczba osobomiesięcy
- Defects/KLOC
- Cost/LOC
- Documentation Pages/KLOC



Zalety i wady LOC

ZALETY

- Łatwa do obliczenia
- Dobra znana i opisana (przed 1979)
- Opiera się na niej wiele modeli prognostycznych

WADY

- Zależy od języka programowania
- Nie promuje krótkich, dobrze zaprojektowanych programów
- Trudno użyć do prognozy – szacowanie LOC przed analizą i projektowaniem

Pomiary funkcjonalności produktu

- Metoda punktów funkcyjnych [Allan Albrecht IBM '79,'83]



- International Function Point Users Group,
www.ifpug.org
- Miara pośrednia
- Wyliczana za pomocą doświadczalnie dobranego wzoru



Punkt funkcyjny (PF)

- Oblicza się na podstawie ważonej sumy wejść, wyjść, zapytań, modyfikacji tabel oraz interfejsów



5 aspektów

- Liczba wejść użytkownika
 - Wprowadza dane, nie są to zapytania
- Liczba wyjść użytkownika
 - Udostępniana informacja (raporty, okienka, komunikaty,...)
 - Nie liczy się oddzielnie poszczególnych elementów, np. na raporcie
- Liczba zapytań użytkownika
 - Prosi o informację i natychmiast uzyskuje ją w postaci wyjścia
 - Każde zapytanie liczy się oddzielnie

5 aspektów

- Liczba plików
 - Wszystkie pliki z danymi
 - Dane różnych rodzajów przechowywane w jednej bazie danych liczy się jako oddzielne pliki
- Liczba interfejsów zewnętrznych
 - Przekazywanie informacji innym systemom



5 aspektów

| Parametr | Liczba | Wagi | | | = | |
|------------|--------|--------|---------|---------|---|--|
| | | Proste | Średnie | Złożone | | |
| Wejście | X | 3 | 4 | 6 | = | |
| Wyjście | X | 4 | 5 | 7 | = | |
| Zapytanie | X | 3 | 4 | 6 | = | |
| Pliki | X | 7 | 10 | 15 | = | |
| Interfejsy | X | 5 | 7 | 10 | = | |
| Suma (S) | | | | | → | |

Obliczenia

$$PF = S \times [0.65 + 0.01 \times \sum(F_i)]$$

- Współczynnik złożoności F_i , $i \in \{1, \dots, 14\}$
 - Ustalany na podstawie odpowiedzi na serię pytań
 - Każda odpowiedź jest liczbą od 0 (nieistotne/nie dotyczy) do 5 (absolutnie kluczowe)



Przykładowe pytania

- (1) Czy system wymaga tworzenia backupów i odzyskiwania danych?
- (3) Czy dane przetwarzane są w sposób rozproszony?
- (10) Czy algorytmy przetwarzania danych są bardzo złożone?
- (12) Czy projekt obejmuje wdrożenie?
- (14) Czy użytkownik ma mieć możliwość modyfikowania systemu. Czy łatwość użytkowania jest ważna?

PF i języki programowania

| <u>Język</u> | <u>AvLOC/ PF</u> |
|-------------------------------|------------------|
| Assembler | 320 |
| C | 128 |
| COBOL | 105 |
| Data base Languages | 40 |
| Objective C | 27 |
| Smalltalk | 21 |
| Graphic icon languages | 4 |

Dreger, J.B., *Function Point Analysis*, Prentice-Hall, 1989, p. 136

PF i języki BD

| <u>Język</u> | <u>AvLOC/ PF</u> |
|------------------------------|------------------|
| Access | 38 |
| INFORMIX | 40 |
| ORACLE | 40 |
| Oracle Developer 2000 | 23 |
| PROGRESS v.4 | 36 |
| SYBASE | 40 |
| ANSI SQL | 13 |

K.Subieta, Budowa i integracja SI

PF i aplikacje

- 1 FP \approx 125 instrukcji w C
- 10 FPs
 - Typowy mały program tworzony samodzielnie przez klienta (1 m-c)
- 100 FPs
 - Większość popularnych aplikacji; wartość typowa dla aplikacji tworzonych przez klienta samodzielnie (6 m-cy)

PF i aplikacje

- 1,000 FPs
 - Komercyjne aplikacje w MS Windows, małe aplikacje klient-serwer (10 osób, ponad 12 m-cy)
- 10,000 FPs
 - Systemy (100 osób, ponad 18 m-cy)
- 100,000 FPs
 - MS Windows'95, MVS, systemy militarne

PF i wydajność zespołu

| <u>Poziom języka</u> | <u>AvPFs/osobomiesiąc</u> |
|----------------------|---------------------------|
| 1 – 3 C | 5 – 10 |
| 4 – 8 C++, Java | 10 – 20 |
| 9 – 15 DELPHI | 16 – 23 |
| 16 – 23 HTML | 15 – 30 |
| 24 – 55 SQL | 30 – 50 |
| >55 | 40 – 100 |

wg. Software Productivity Research

Wykorzystanie PF

- Ocena złożoności realizacji systemów
- Audyt projektów
- Wybór systemów informatycznych funkcjonujących w przedsiębiorstwie do reinżynierii (wg. koszt utrzymania/FPs)
- Szacowanie liczby testów
- Ocena jakości pracy i wydajności zespołów ludzkich

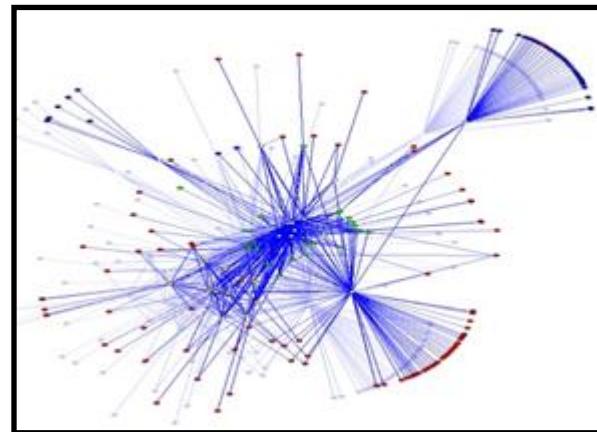
Wykorzystanie PF

- Ocena stopnia zmian, wprowadzanych przez użytkownika na poszczególnych etapach budowy systemu informatycznego
- Prognozowanie kosztów pielęgnacji i rozwoju systemów
- Porównanie i ocena różnych ofert dostawców oprogramowania pod kątem merytorycznym i kosztowym



Pomiary złożoności produktu

- Nauka o programach Halsteada
 - Halstead's Software Science
- Liczba cyklotomyczna McCabe'a
 - McCabe's cyclomatic number



sbs-xnet.sbs.ox.ac.uk/complexity/

Nauka o programach Halsteada

- Halstead(1977) : Elements of Software Science, New York, Elsevier North-Holland
 - yunus.hun.edu.tr/~sencer/complexity.html
- Bazuje na elementach składniowych programu
 - operatorach
 - operandach



Nauka o programach Halsteada

- n_1 - liczba różnych operatorów
 - Operatory to: "+", "*", "[...]", ";" ...
- n_2 - liczba różnych operandów
 - Operandy to: stałe, zmienne, wyrażenia
- N_1 - całkowita liczba wystąpień operatorów
- N_2 - całkowita liczba wystąpień operandów

Przykład

- Różne operatory

- if () { } > < = * ;

- $n_1 = 10$

- $N_1 = 13$

```
if (k < 2) {  
    if (k > 3)  
        x = x*k;  
}
```

- Różne operandy

- k 2 3 x

- $n_2 = 4$

- $N_2 = 7$

Metryka Halsteada

- Weryfikowana eksperymentalnie
- Słownik: $n=n_1+n_2$
- Długość implementacji: $N=N_1+N_2$
- Volume: $V = N \log_2 n$
 - Liczba bitów niezbędna do zakodowania elementów słownika
- Programming Effort: $E=V/L$
 - $L = V^*/V$
 - V^* rozmiar najbardziej zwięzzej implementacji

Metryka Halsteada - ZALETY

- Nie wymaga wnikliwej analizy kodu
- Prognozuje stopień trudności utrzymania
- Pomaga w planowaniu harmonogramu i raportowaniu
- Mierzy ogólną jakość programu
- Prosta do wyliczenia
- Może być użyta dla dowolnego języka programowania
- Doświadczalnie wykazano jej przydatność w predykcji wysiłku (programming effort) i oczekiwanej liczby błędów programistycznych

Metryka Halsteada - WADY

- Zależy od finalnej postaci kodu
- Nie nadaje się do wykorzystania na etapie projektowania (modelu)



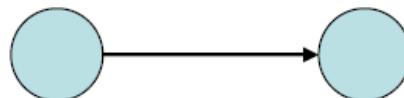
Liczba cyklotomatyczna McCabe'a

- Oparta na schemacie blokowym programu i liczbie niezależnych przebiegów
- Przebiegi reprezentowane są za pomocą diagramu sterującego (grafu)
 - Węzły reprezentują sekwencyjne bloki kodu
 - Krawędzie reprezentują możliwe ścieżki, zależne od decyzji

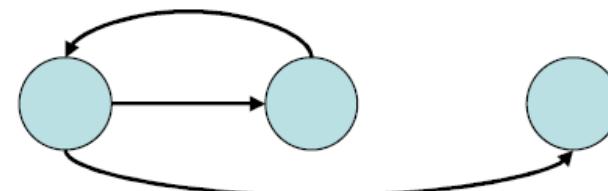


Flow Graph Notation

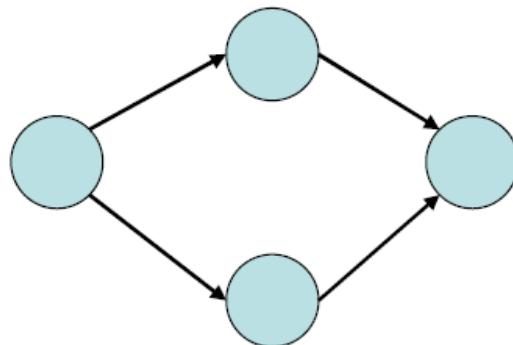
Sequence



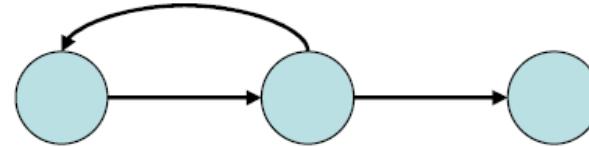
While



If-then-else



Until



Jonathan I. Maletic, Software Metrics



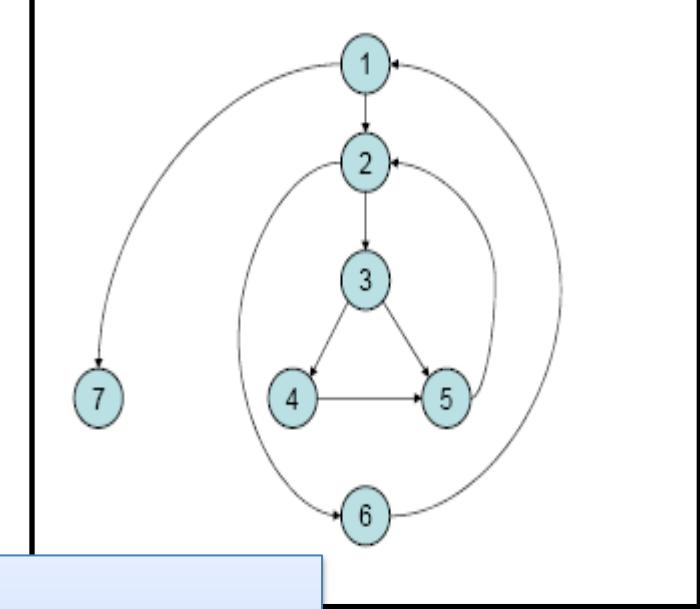
Liczba cyklowatyczna

- Zbiór niezależnych ścieżek w grafie
- $V(G) = E - N + 2$
 - E liczba krawędzi
 - N liczba węzłów
- $V(G) = P + 1$
 - P liczba węzłów warunkowych



Przykład

```
i = 0;  
while (i<n-1) do  
    j = i + 1;  
    while (j<n) do  
        if A[i]<A[j] then  
            swap(A[i], A[j]);  
        end do;  
        i=i+1;  
    end do;
```



$$V(G) = 9 - 7 + 2 = 4$$

$$V(G) = 3 + 1 = 4$$

Zbiór przebiegów:

- 1, 7
- 1, 2, 6, 1, 7
- 1, 2, 3, 4, 5, 2, 6, 1, 7
- 1, 2, 3, 5, 2, 6, 1, 7

Interpretacja

- $V(G)$ jest liczbą regionów wyznaczonych przez graf planarny
- Liczba regionów wzrasta wraz z liczbą ścieżek warunkowych i pętli
- Liczbową miara trudności testowania
- Doświadczenia pokazują, że $V(G)$ nie powinna przekraczać 10
 - Dla większej wartości testowanie jest bardzo trudne

Liczba cykloomatyczna – ZALETY

- Do mierzenia stopnia trudności pielęgnacji
- Miara jakości – porównanie różnych projektów
- Może być wcześniej obliczona niż miara Halsteada
- Prosta do wyliczenia



Liczba cykematyczna – WADY

- Mierzy złożoność wątków kontroli, a nie złożoność danych
- Pętle zagnieżdżone mają tą samą miarę, co nie zagnieżdżone
 - Zagnieżdżone struktury warunkowe są trudniejsze do zrozumienia
- Może dawać mylne wyobrażenia dotyczące prostych porównań i struktur warunkowych



Obiekty i metryki

- Przyczyny różnic pomiędzy metrykami OIO, a IO
 - Lokalizacja
 - Enkapsulacja
 - Ukrywanie informacji
 - Dziedziczenie
 - Obiektowe techniki abstrakcyjne



Obiekty i metryki - przykłady

- DIT (Depth of Inheritance Tree)
 - Głębokość drzewa dziedziczenia obiektów
- NOC (Number of Children)
 - Liczba potomków w ramach dziedziczenia dla konkretnej klasy
- CBO (Coupling Between Object Classes)
 - Liczba obiektów, dla których analizowany obiekt jest łącznikiem do innych obiektów

Obiekty i metryki - przykłady

- RFC (Response for a Class)
 - Liczność zbioru metod (lokalnych i z innych obiektów) wywoływanych przez metody obiektu
- LCOM (Lack of Cohesion in Methods)
 - Liczba metod nie wykorzystywanych przez inne metody
- WMC (Weighted methods per class)
 - Rozmiar metod w konkretnych klasach: zarówno liczba metod, jak i ich złożoność (liczona na przykład tradycyjnymi metodami dla programów nieobiektowych).

OO metryki dla hermetyzacji

- PCTPUB (Percent Public Data) **HERMETYZACJA!**
 - Procent danych publicznych; liczony jako stosunek danych publicznych względem danych prywatnych w obiektach
- PUBDATA (Access to Public Data)
 - Dostępność danych publicznych; określany miarą liczby dostępów do danych publicznych

OO metryki dla polimorfizmu

- PCTCALL (Percent of Unoverloaded Calls)
 - Stosunek liczby nieprzeciążonych wywołań metod do wszystkich wywołań
- ROOTCNT (Number of Roots)
 - Globalna liczba korzeni w hierarchii dziedziczenia klas
- FANIN (Fan-in)
 - Liczba klas, z których wywodzi się analizowana klasa



OO metryki dla projektu

- MAXV (Maximum $v(G)$)
 - Maksymalna wartość złożoności (tzw. cyklotomatycznej) dla metod
- MAXEV (Maximum $ev(G)$)
 - Maksymalna wartość złożoności struktury kodu dla metod
- QUAL (Hierarchy Quality)
 - Liczba klas uzależnionych od poprawnego funkcjonowania swoich przodków

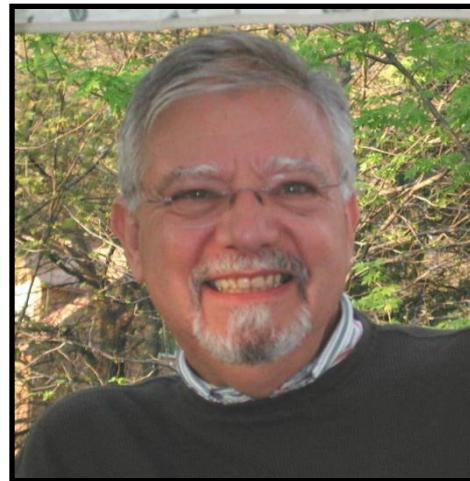
Jak i co mierzymy? Program!

- Metryki i gromadzenie danych
- Jak się do tego zabrać
 - Quality Function Deployment
 - Goal Question Metric
 - Software Quality Metrics



GQM

- Goal – Question – Metric
 - Victor R. Basili, Maryland University, 1988
 - NASA Software Engineering Laboratory (SEL),

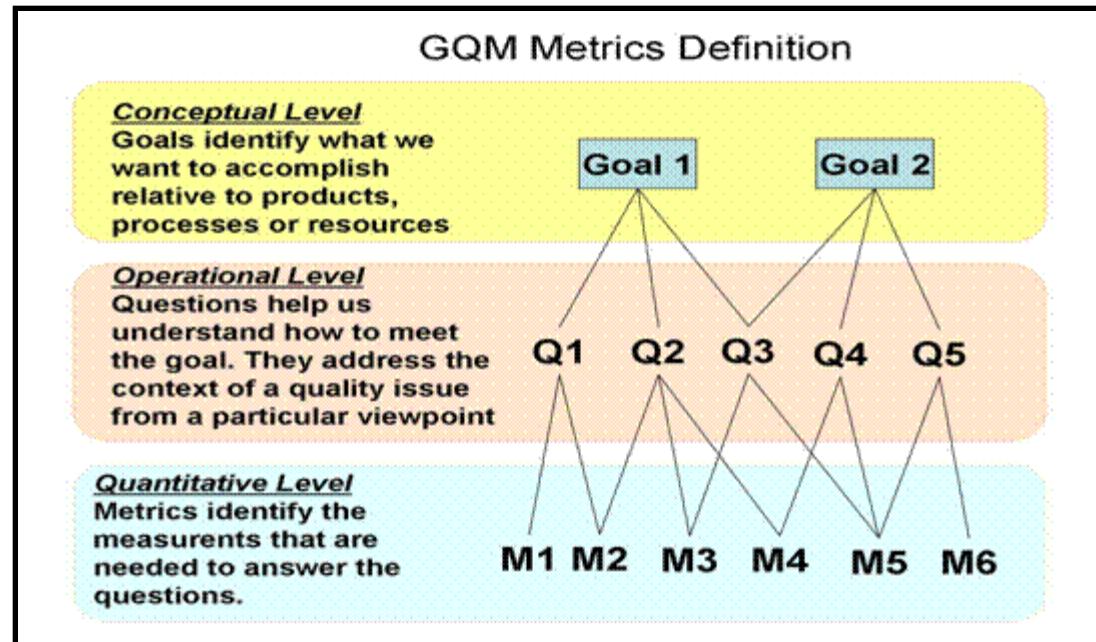


GQM paradigm

- Generate a set of goals
- Derive a set of questions
- Develop a set of metrics



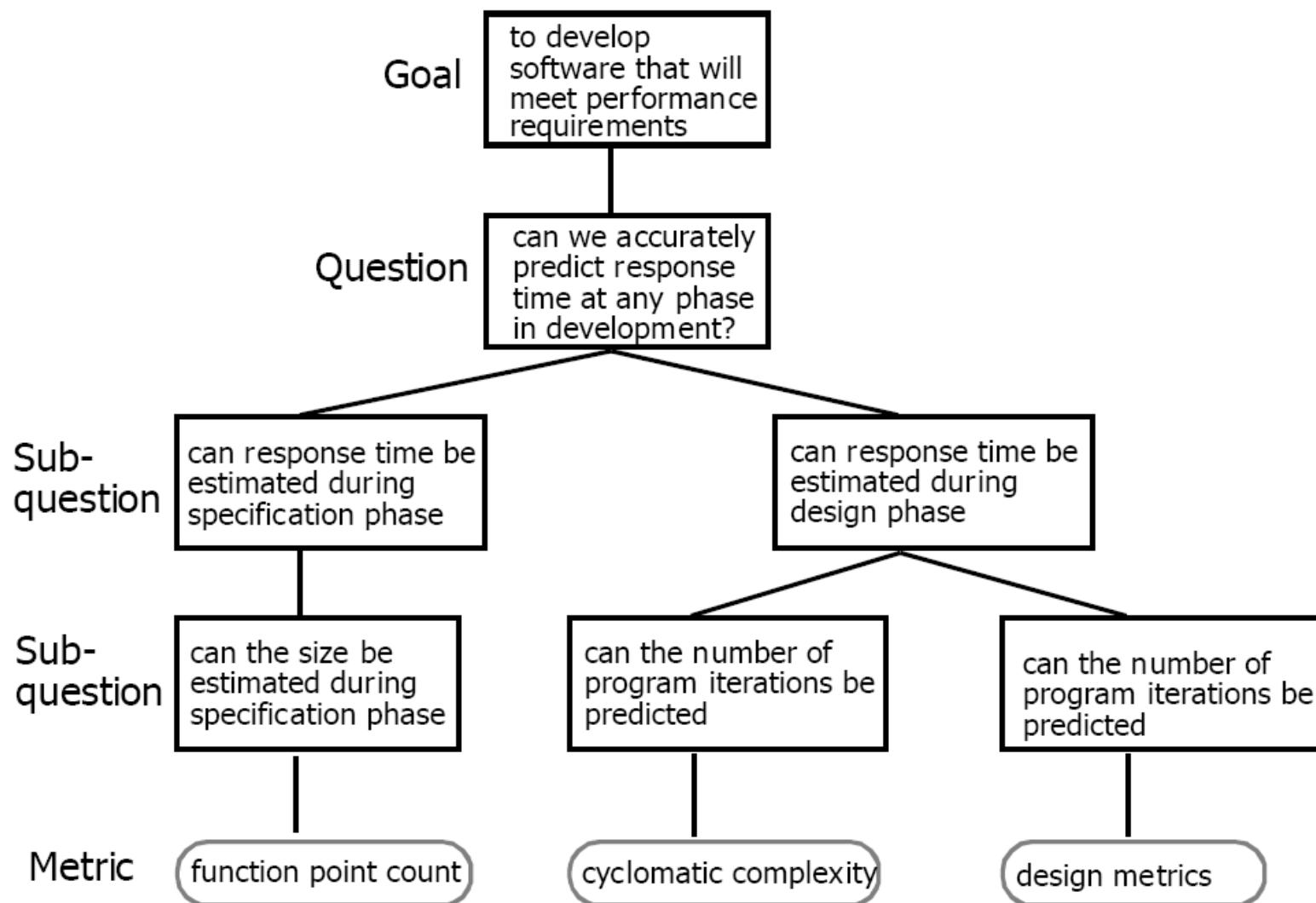
Basic GQM Hierarchy



<http://www.goldpractices.com/practices/gqm>

GQM example

Anthony Finkelstein



Gromadzenie danych

- Konieczne do pomiarów
- Są gotowe narzędzia
 - Krakatau: C/C++, Java
 - Cyclomatic Complexity, Halstead Software Science metrics, LOC metrics; ponad 70 metryk



Etykieta pomiarów kierownika

- Interpretując wyniki, kieruj się zdrowym rozsądkiem
- Udostępnij wyniki pracownikom i zespołom, którzy je prowadzą
- Nie oceniaj poszczególnych pracowników na podstawie wyników pomiarów
- Wspólnie z pracownikami zastanów się, po co pomiary i dobierz metryki do celów

Robert Grady

Etykieta pomiarów kierownika

- Nigdy nie używaj wyników pomiarów, aby grozić pracownikom
- Wyniki wskazujące na pojawiające się trudności nie są ZŁE – traktuj je jako wskazówkę, co można poprawić
- Nie przywiązuj nadmiernej uwagi do jednej miary, zaniedbując inne

Robert Grady

Uwagi

- Stosowanie tylko jednej metryki jest mało przydatne
- Najlepszy zestaw metryk nie musi być od razu znany
 - dlatego na początku można stosować nawet kilkadziesiąt różnych technik
- Docelowo najlepiej korzystać z 3 - 5 dobrze przemyślanych metryk
- Metryki są prawie zawsze ze sobą powiązane
 - Stosowanie jednej implikuje użycie drugiej

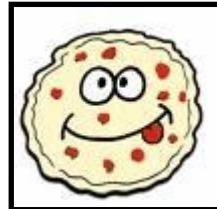
M. Trzaska, Metryki w obiektowej io

Uwagi

- Metryki powinny być stosowane w sposób regularny, w miarę możliwości automatyczny
- Wykorzystywane metryki NIE mogą wpływać na mierzony element
- Warto liczyć średnią wartość metryki dla całego elementu i badać lokalne odchylenia
- Metryki mogą być **SZKODLIWE**... tylko wtedy gdy je **ŽLE STOSUJEMY**

Metryki z życia wzięte 😊

- The Pizza Metric
 - **Jak:** Policz liczbę pudełek po pizzy.
 - **Co:** Mierzy stopień niedoszacowania harmonogramu. Jeśli ludzie muszą pracować po godzinach, to gdzieś popełniono błąd w oszacowaniu czasu realizacji projektu.



Metryki z życia wzięte 😊



- The Beer Metric
 - **Jak:** Zapraszaj zespół co piątek na piwo i notuj wysokość rachunków.
 - **Co:** Podobnie jak the Aspirin Metric, mierzy poziom frustracji zespołu. Między innymi, może świadczyć o tym, że problemy techniczne przerosły oczekiwania.
- <http://www.elsop.com/wrc/humor/softmetr.htm>

KONIEC

