Nickname: mike6606
Age: 8 years 10 months
Fans: 0
Followers: 5
+ Plus follow

< March 2023 >

day	One	Two	Three	Four	Five	Six
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Search

Look for it

Google search

Frequently used links

My essays
My comment
My involvement
Latest comments
My labels

Essay archives

January 2021 (1)
January 2018 (3)
January 2017 (8)

Read the leaderboard

1. Software Engineering Practitioners' Research Methods Chapter 13 Answers (1578)
2. Software Engineering Practitioners' Research Methods Chapter 1471 Answers (<>)
3. Software Engineering Research Methods for Practitioners Chapter 1290 Answers (<>)
4. Software Engineering Practitioners' Research Methods Chapter 34 Answers (1282)
5. Software Engineering Practitioners' Research Methods Chapter 1264 Answers (<>)

Recommended leaderboards

1. Software Engineering Practitioners' Research Methods Chapter 1 Answers (<>)

Software Engineering Practitioners' Research Methods Chapter 30 Answers

Problem:

Measurement theory is an advanced topic that has a strong bearing on software metrics. Using [Zus97], [Fen91], [Zus90] or Web-based sources, write a brief paper that outlines the main tenets of measurement theory. Individual project: Develop a presentation on the subject and present it to your class.

Answer:

4633-23-1P SA: 9420

SR: 6376

Measurement theory is a very convenient theoretical framework to define the underlying theories upon which software engineering measures are based. Measurement theory is not shared by many statisticians and data analysts. The major terms in the measurement theory are:

• **Measurement** is the process by which numbers or symbols are assigned to the attributes of entities in the real worl in such a way as to define them according to clearly defined rules. Measurement is the act of determining a measure.

• A **measure** provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute o a product or process.

• A **metric** is a quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Main tenets (principles) of Measurement Theory:

A measurement process can be characterized by five activities. They are:

1. **Formulation:** The derivation of software measures and metrics appropriate for the representation of the software that is being considered.

2. **Collection:** The mechanism used to collect data required to derive the formulated metrics.

3. **Analysis:** The computation of metrics and the application of mathematical tools.

4. **Interpretation**: The evaluation of metrics results in an effort to gain insight into the quality of the representation.

5. **Feedback:** Recommendations derived from the interpretation of product metrics transmitted to the software team.

Software metrics will be useful only if they are characterized effectively and validated so that their own worth is proven. The following principles are representative of many that can be proposed for metrics characterization and validation.

• A metric should have desirable mathematical properties. That is, the metric' s value should be in a meaningful range also, a metric that purports to be on a rational scale should not be composed of components that are only measured on an ordinary scale.

• When a metric represents a software characteristic that increases when positive traits occur or decreases when undesirable traits are encountered, the value of the metric should increase or decrease in the same manner.

• Each metric should be validated empirically in a wide variety of context before being published or used to make decisions. A metric should measure the factor of interest, independently of other factors. It should "scale up" to large systems and work in a variety of programming languages and system domains.

Problem:

Why is it that a single, all-encompassing metric cannot be developed for program complexity or program quality? Try to come up with a measure or metric from everyday life that violates the attributes of effective software metrics defined in Section 30.1.5.

Answer:

4633-23-2P SA: 9420

SR: 6376

A single, all –encompassing metrics cannot be developed for program complexity or quality because, even though hundreds of metrics have been proposed for computer software, not all of them provide practical support to the software engineer. Some demand measurement, that is too complex, others are so doubtful that few real-world

Although most software metrics satisfy the attributes of effective software, some commonly used metrics may fail to satisfy one or two of the attributes. One such commonly used (everyday life) metric is the **function point** metric. It is said that the function point metric fails to satisfy the objective – consistent and objective.

Problem:

A system has 12 external inputs, 24 external outputs, fields 30 different external queries, manages 4 internal logical files, and interfaces with 6 different legacy systems (6 EIFs). All of these data are of average complexity and the overal system is relatively simple. Compute FP for the system.

Answer:

Consider the following data:

The system has external inputs (EI' s) =12

External outputs (EO' s) = 24

External queries (EQ' s) = 30

Internal logical files (ILF' s) = 4

External interface files (EIF' s) = 6

All of these data are of average complexity = 3.

Note:

Ordinal scale ranges from 0(applicable) to 5 (Essential):

0 - No Influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 – Essential

Functioning point metric:

Information Domain value	Weighting factor			
	count	simple	Average	Complex
External inputs (EI's)	12 ×	3	4	6 = 36
External output (EO's)	24 ×	4	5	7 = 96
External inquiries (EQ's)	30 ×	3	4	6 = 90
Internal logical files	4 ×	7	10	15 = 28
External interface files	6 ×	5	7	10 = 30
Count total				280

To compute function points:

$$Fp = \text{count total} \times \left[0.65 + 0.01 \times \sum(F_i)\right]$$

To find $F_i (i = 1 \text{ to } 14)$ are value adjustment factors(VAF) based on the responses. But as per the scenario, all of these data are of average complexity.

$$\sum(F_i) = 14 \times 3$$
$$= 42$$

Since, value of $\sum(F_i)$ is 42.

$$Fp = \text{count total} \times \left[0.65 + 0.01 \times \sum (F_i)\right]$$
$$= 280 \times \left[0.65 + 0.01 \times 42\right]$$
$$= 280 \times \left[0.65 + 0.42\right]$$
$$= 299.6$$

Therefore, the functional points(FP) is 299.6.

Problem:

Software for System X has 24 individual functional requirements and 14 nonfunctional requirements. What is the specificity of the requirements? The completeness?

Answer:

Soft are for system X has 24 individual functional requirements and 14 non functional requirements.

$n_r = n_f + n_{nf}$

$n_f =$ functional requirements

$n_{nf} =$ Non – functional requirements

$n_r = 24 + 14$

$= 38$

To determine the specificity (lack of ambiguity) of requirements

A metric that is based on the consistency of the reviewers interpretation of each requirement.

$Q_1 = n_{ui} / n_r$

Where n_{ui} is the number of requirements for which all reviewers had identical interpretations. The closer the value of Q_1 to 1, the lower is the ambiguity of the specification.

$1 = n_{ui} / 38$

$n_{ui} = 38$

The completeness of functional requirements can be determined by computing the ratio

$Q_2 = n_u / \left[n_i \times n_s\right]$

$= 24 / \left[n_i \times n_s\right]$

Where n_u is the number of unique function requirements, n_i is the number of inputs defined or implied by the specification and n_s is the number of states specified.

Problem:

A major information system has 1140 modules. There are 96 modules that perform control and coordination function and 490 modules whose function depends on prior processing. The system processes approximately 220 data objects that each have an average of three attributes. There are 140 unique database items and 90 different database segments. Finally, 600 modules have single entry and exit points. Compute the DSQI for this system.

Answer:

The information system obtained from data and architectural design to derive a design structure quality index (DSQI) that ranges from 0 to 1.

$S_1 =$ The total number of modules defined in the information system.

$S_2 =$ The number of modules whose correct function depends on the source of data input or that produce data to b used else

$S_3 =$ The number of modules whose correct function depends on prior processing.

$S_5 =$ The total number of unique data base items.

$S_6 =$ The number of data base segments

$S_7 =$ The number of modules with a single entry and exit.

Once values S_1 through S_7 are determined for a computer program, the following intermediate value can be computed.

Program structure: D_1 where D_1 is defined as follows: if the architectural design was developed using a distinct method (e.g data flow oriented design or object – oriented design).Then $D_1 = 1$ otherwise $D_1 = 0$.

Module independence : $D = 1 - (S_2 \mid S_2)$

Module not dependent on prior processing : $D_3 = 1 - (S_3 \mid S_1)$

Database size : $D_4 = 1 - (S_5 \mid S_4)$

Database compartmentalization : $D_5 = 1 - (S_6 \mid S_4)$

Module entrance / exit characteristic : $D_6 = 1 - (S_7 \mid S_1)$

With these intermediate values determined, the DSQI is computed in the following manner:

$$DSQI = \sum W_i D_i$$

Where i = 1 to 6, W_i is the value relative weighting of the importance of each of the intermediate values and $\sum W_i = 1 (\text{if all } D_i \text{ are weighted equally, then } W_i = 0.167)$

$S_1 = 1140 \qquad S_5 = 140$

$S_2 = 96 \qquad S_6 = 90$

$S_3 = 490 \qquad S_6 = 600$

$$\begin{aligned} S_4 &= (220 \times 3) + 220 \\ &= 660 + 220 \\ &= 880 \end{aligned}$$

$D_1 = 0$

Module independence : $D_2 = 1 - (S_2 \mid S_1)$

$= 1 - (96 \mid 1140)$

$= 1 - 0.08$

$= 0.92$

Modules not independent on prior processing : $D_3 = 1 - (S_3 \mid S_1)$

$= 1 - (490 \mid 1140)$

$= 1 - 0.42$

$= 0.58$

Database size : $D_4 = 1 - (S_5 \mid S_4)$

$= 1 - (140 \mid 880)$

$= 1 - 0.15$

Database compartmentalization: $D_5 = 1-(S_6 \mid S_4)$

$=1-(90 \mid 880)$

$=1-0.1$

$=0.9$

Module entrance/ exit characteristic: $D_6 = 1-(S_7 \mid S_1)$

$=1-(600 \mid 1140)$

$=1-0.52$

$=0.48$

The DSQI is computed in the following manner

Assuming wi=0.167,

DSQI = 0.167 SUM (Di)

=0.644

Problem:

A class **X** has 12 operations. Cyclomatic complexity has been computed for all operations in the OO system, and the average value of module complexity is 4. For class **X**, the complexity for operations 1 to 12 is 5, 4, 3, 3, 6, 8, 2, 2, 5, 5, 4, 4, respectively. Compute the weighted methods per class.

Answer:

WMC is simply the sum of each of the individual complexities divided by the average complexity:

Average complexity is =4

WMC=(5/4)+(4/4)+(3/4)+(3/4)+(6/4)+(8/4)+(2/4)+(5/4)+(5/4)+(4/4)+(4/4)

WMC = 1.25 + 1 + 0.75 + 0.75 + 1.5 + 2 + 0.5 + 0.5 + 1.25 + 1.25 + 1 + 1

= 12.75

Problem:

Develop a software tool that will compute cyclomatic complexity for a programming language module. You may choose the language.

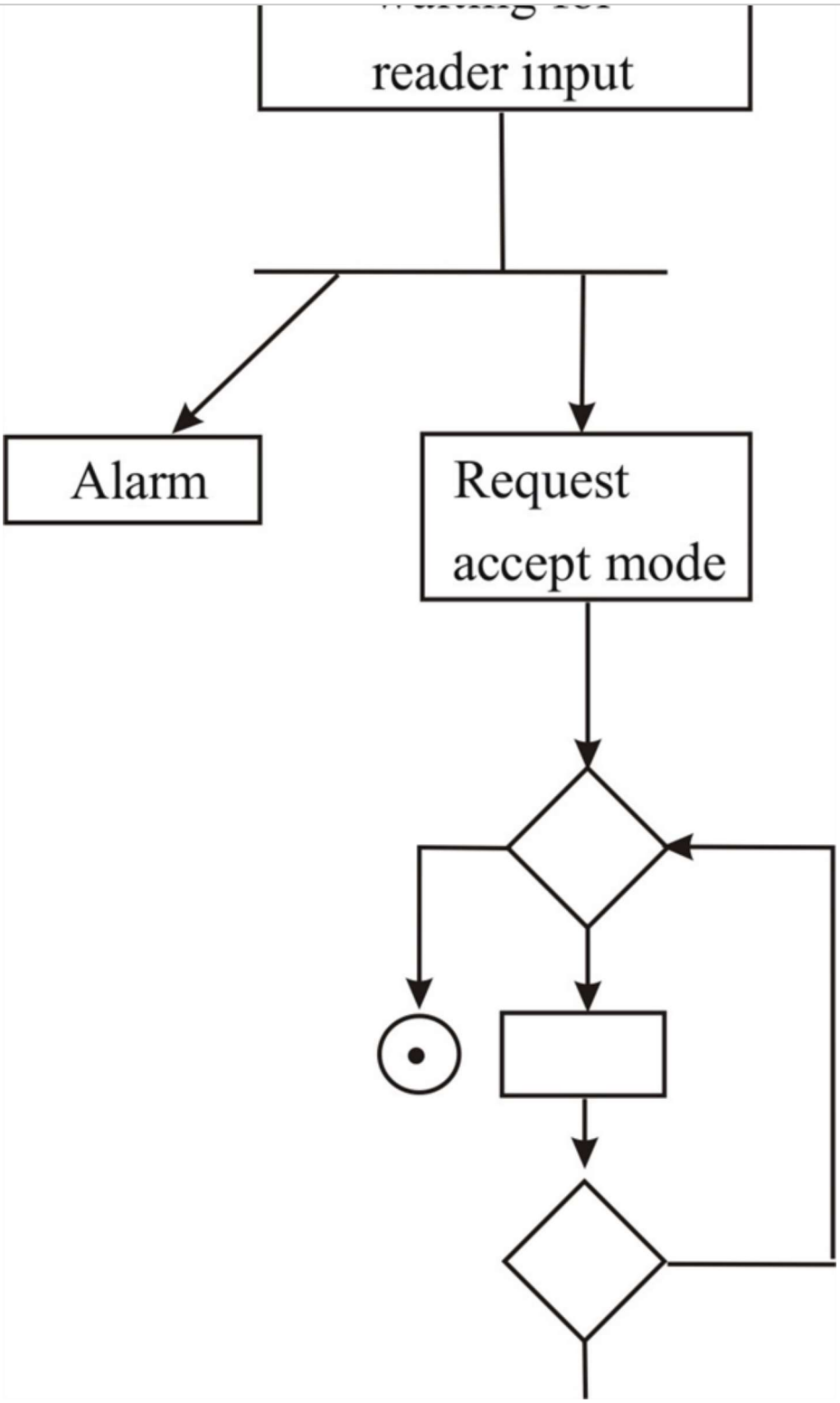
Answer:

Cyclomatic complexity for a programming language module:-

Cyclomatic complexity is software metric and it is used to measure the complexity of a program. It directly measures the number of linearly independent path through a program' s source code.

Consider the sensor software. It is an automated tool for identify the users in login process.

Flow diagram of this tool:-



In this process, the reader input waiting for user’ s request.

If the first request is valid then control goes to another one and if the first request is not valid then loop go back for valid request. The process is goes like this.

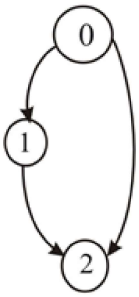
According this tool, the cyclomatic metric measures the amount of decision logic in a single software module. In my implementation, I have taken the software module to be a function. The cyclomatic complexity is based entirely on the Structure of the software’ s control flow graph.

Formula for calculating the cyclomatic complexity for each function using it’ s control flow graph is

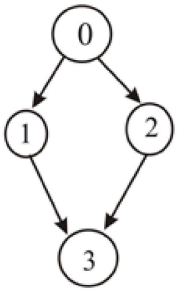
$$E = N + 2$$

are the number of edges and nodes

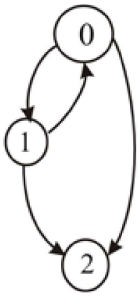
Control flow graphs for the control structures found in C++ language except for 'for loop'



1f

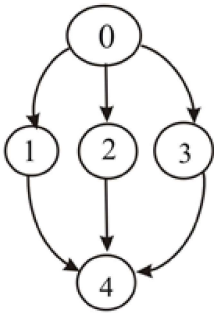


if/else



while

do



switch

Control flow graphs that are used in our sensor tool designing section

Problem:

Develop a small software tool that will perform a Halstead analysis on programming language source code of your choosing.

Answer:

592-15-10P SA CODE: 4475

SR CODE: 4467

Consider the software tool, that calculate the two integer numbers

Source code of adding two integers in Java

Public class Add numbers

{

Public static void main (string [7 args)

{

System.out.print ln ("Adding of two numbers!");

int a = Integer. Parse Int (args [0]);

int b = Integer. Parse Int (args [1]);

int sum = a+b;

system. Out. Print ln ("sum:" + sum);

}

}.

Now take the Halstead' s theory and based on analysis of this theory through above source code we may observe like this.

Hallstead uses the primitive measures to develop expressions for the overall program length, potential minimum volume for an algorithm, the actual volume, the language level and other features such as development effort,

$$N=n.\log_2 n_1+n_2\log_2 n_2$$

Program volume (V) may defined

$$V=N\log_2(n_1+n_2)$$

And

Volume ratio (L) expressed as

$$L=2/n_1\times n_2/N_2$$

Here

$n_1 \rightarrow$ The number of distinct operators that appear in a program

$n_2 \rightarrow$ The number of distinct operands that appear in a program

$N_1 \rightarrow$ Total number of operator occurrences

$N_2 \rightarrow$ Total number of operand occurrences

So, from the above source code

$n_1=1$ (that is 't')

$n_2=2$ (that is variable 'a & b')

We apply these values to Halstead analysis theory. Than

$$N=1\log_2 1+2\log_2 2$$

$$0+0.602$$

So $N=0.602$

Now we need to calculate the volume (v)

$$V=0.602\log_2(1+2)$$

$$=0.602\log_2(3)$$

$$=0.287$$

Now we need to calculate the volume ratio (L)

$$\Rightarrow \quad L=\frac{2}{1}\times\frac{1}{2}=1$$

$$\Rightarrow \quad L=1$$

Like this we analysis the program source code using Halstead analysis

Problem:

A legacy system has 940 modules. The latest release required that 90 of these modules be changed. In addition, 40 new modules were added and 12 old modules were removed. Compute the software maturity index for the system.

Answer:

A software maturity index (SMI) that provides an indication of the stability of a software product (based on changes that occur for each release of the product).

$M_T =$ The number of modules in the legacy system

$F_c =$ The number of modules in the latest release that have been changed

$F_a =$ The number of modules in the latest release that have been added.

$F_d =$ The number of modules from the preceding release that were removed in the latest

$F_c = 90$

$F_a = 40$

$F_d = 12$

The software maturity index is computed in the following manner:

$$SMI = \left[M_T - \left(F_a + F_c + F_d \right) \right] / M_T$$

$$= \left[940 - \left(40 + 90 + 12 \right) \right] / 940$$

$$= 798 / 904$$

$$= 0.848$$

Solution: CHAPTER30: PRODUCT METRICS

30.1 As an alternative, you might assign readings from one or more recent papers on the subject.

30.2 That' s like asking for a single metric to describe a human being. Should it be weight, height, eye color, IQ, EQ, foot size, ... Software systems are just too complex and have too many attributes, all of which are meaningful. Therefore, the search for a single number is fruitless.

30.3 Unadjusted count total = 340, questionnaire total = 3 * 14 = 42, FP = 363.8 (using one of the tiny tools on the SEPA website.

30.4 The specificity cannot be computed without making assumptions about the number of requirements on which reviewers have identical interpretations for (e.g. if reviewers agree on all 38 requirements Q1 = 38/38 = 1 Completeness cannot be computed without making some assumptions on the number of requirements that have been validated as correct and how many have not yet been validates (e.g. if all 38 have been validated the Q3 = 38/(38 + 0) = 1).

30.5. To compute DSQI, we use equation and related definitions:

S1 = 1140

S2 = 1140 - 96 = 1044

S3 = 490

S4 = 220 + 220*3 = 880

S5 = 140

S6 = 90

S7 = 600

we assume D1 = 1

D2 = 1 - (1044/1140) = 0.08

D3 = 1 - (490/1140) = 0.57

D4 = 1 - (140/880) = 0.84

D5 = 1 - (90/880) = 0.90

D6 = 1 - (600/1140) = 0.47

Assuming wi = 0.167,

DSQI = 0.167 SUM (Di)

= 0.644

This value has no meaning in a vacuum. It must be compared to values for other applications. If lower, desigr may be suspect.

30.6 WMC is simply the sum of each of the individual complexities divided by the average complexity:

= 12.75

30.7 – 30.8 These exercises could be good term projects (depending on the nature of the input to the tools).

30.9 The software maturity index is computed using equation

Mt = 940

Fc = 90

Fa = 40

Fd = 12

SMI = [940 - (40 + 90 + 12)] / 940 = 0.85

This value indicated that the system has a way to go before it fully stabilizes (SMI -> 1.0).

+

好文要顶 关注我 收藏该文

mike6606
粉丝 - 0 关注 - 5

0 0

+加关注

« 上一篇： 软件工程 实践者的研究方法 第28章答案
» 下一篇： 软件工程 实践者的研究方法 第31章答案

posted @ 2021-01-22 00:54 mike6606 阅读(241) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页

【阿里云】2核2G云服务器低至99元/年，百款云产品优惠享不停

编辑推荐：
· 现代图片性能优化 - 图片资源的容错及可访问性处理
· 浅谈：服务架构进化论
· 我又和 redis 超时杠上了
· 巧用 CSS 变量，制作高级感拉满的网格动画
· 记录一次锁的优化

阅读排行：
· 使用 Vue 3 时应避免的 10 个错误
· ASP.NET Core Web API 接口限流
· 【故障公告】cc攻击又来了，雪上加霜的三月
· 现代图片性能优化及体验优化指南 - 图片资源的容错及可访问性处理
· 如何在 Net6.0 中对 WebAPI 进行 JWT 认证和授权

Copyright © 2023 mike6606
Powered by .NET 7.0 on Kubernetes