

1. Kto meria úspešnosť projektu?

Vo finalnom dosledku len používateľ alebo zakazník vedia určiť uspešnosť projectu. Urobiť spravne finalne testovanie a vyhodnotiť uroveň uspešnosť projectu.

2. Čo je metrika?

Metrika je výsledkom merania softvérového produktu, softvérového procesu a t.d. Reprezentuje sebou nejaký počet riadkov kodu, commentarov, klasov, metod a t.d.

3. Dve triedy softvérových metrík

Produkty a procesy. Produkty reprezentuju pripravený programy, documentacie, reporty. Procesy, ktorými je produkt vyvíjaný (fázy životného cyklu softvéru, špecifikácia, návrh, implementácia, testovanie).

4. Čo je meranie?

Merania sú pozorovania vykonané starostlivo. Merania sa používajú na identifikáciu nových javov (prieskumné merania), testovanie predpokladaných javov, alebo návod na inžinierstvo pri aplikácii modelov. Každé meranie vyžaduje mierku alebo metriku.

5. Matematická notácia pre metriky

6. Externé metriky produktu

Externé metriky produktu pokrývajú vlastnosti viditeľné pre používateľov produktu. Metriky nespoľahlivosti produktu, funkčnosti, výkonu, použiteľnosti, nákladové metriky.

7. Interné metriky produktu

Interné metriky produktu pokrývajú vlastnosti viditeľné iba pre vývojový tím. Metriky veľkosti, zložitosti, štýlu.

8. Metriky projektu

Lines of Code, McCabe, Henry-Kafura, Function points

9. LOC

Použitie počtu riadkov pochádza z programu Assembler, v ktorom je riadok kódu a príkaz inštrukcia to isté. Existuje tri variacie: fyzické riadky kódu, logicke a riadky s komentármami.

10. Cyklometrická zložitosť (všeobecne)

Označuje testovateľnosť a udržovateľnosť programu. Pri aplikácii na softvér je to počet lineárne nezávislých cest, ktoré tvoria program. Ako taký ho možno použiť na označenie úsilia potrebného na testovanie programu.

11. Cyklometrická zložitosť sekvencie

12. Cyklometrická zložitosť vetvenia

13. Cyklometrická zložitosť cyklu

14. Fan-in a Fan-out

Fan-in: Počet modulov, ktoré volajú daný modul. Fan-out: Počet modulov, ktoré sú volané daným modulom. Vo všeobecnosti sú moduly s veľkým Fan-in relatívne malé a jednoduché a zvyčajne sa nachádzajú v spodných vrstvách konštrukcie. Naproti tomu moduly, ktoré sú veľké a zložité, budú mať pravdepodobne malý Fan-in. Preto moduly alebo komponenty, ktoré majú veľký Fan-in a veľký Fan-out, môžu naznačovať zlý dizajn.

15. FPA - základný princíp

Function points analysis. Sledovanie funkčného tečenia, hodnotenie a uprednostňovanie práce.

16. FPA - ako na to?

Zdroje a rozpočet na podporu plánovania, benchmarking, identifikácia osvedčených postupov, plánovanie nových vydaní, hodnotenie softvérového majetku. Hlavne výhody: pôvodne pre odhad nákladov. Dá sa použiť v ktorejkoľvek fáze životného cyklu softvéru.

17. COCOMO 81

Constructive Cost Model. Model používa základný regresný vzorec s parametrami, ktoré sú odvodené z historických údajov projektu a súčasných, ako aj budúcich charakteristík projektu.

COCOMO 81 – Waterfall

Basic, Intermediate, or Detailed for model names

Organic, Semidetached, or Embedded for development mode

18. COCOMO II

Model names are Application Composition, Early Design, or Post-architecture; Scale factors: Precedentedness (PREC), Development Flexibility (FLEX), Architecture/Risk Resolution (RESL), Team Cohesion (TEAM), and Process Maturity (PMAT).
modern SW processes

19. Code-and-fix model

Tento model je najjednoduchší, je založený na tom, že pri kódovaní sa hľadajú chyby a tento cyklus sa opakuje až do dokončenia projektu.

20. Vodopádový model

Model vodopádu je lineárny, sekvenčný prístup k životnému cyklu vývoja softvéru. Pozostáva z niekoľkých častí: najprv ide Requirement gathering and analysis, System design, Implementation, Testing, Deployment of system, Maintenance

21. Vylepšený vodopádový model

Má viacero atribútov a podmienok. System requirements, Software requirements, Predbežné program design, Analysis, Program design, Testing, Operations.

22. UP

Unified process. Je iteratívny a postupný vývojový process. Pozostáva z Use case model, Analysis model, Design model, Deployment model, Implementation model, Test model.

23. Cleanroom

24. V-model

25. Špirálový model

26. Scrum

-Je to metoda na rozdelenie a kontrolu uloh. Rozdeľuje prácu na určité stupne: vlastník, scrum master, tím, ľudi.
-Má svoj životný cyklus, zväčša sa iteruje v 2-4 týždenných iteráciách s každodennými stretnutiami, je tam nastavenie cieľov na začiatku, potom vývoj, na konci rekapitulácia čo sa stihlo a čo nie prečo atď...

27. RAD

28. DSDM

29. XP, párové programovanie

30. TDD

31. FDD

32. Prototypovanie

33. DAD

34. Povedomie (agilného) tímu

35. AMEISE

architektúra klient – server, kde server je simulačný prostriedok, a klient je v interakcii so simulačným prostriedkom. Je to dedičný systém kde možem skusiť realný management projektu.

36. Manažment špecifikácie a skorého návrhu

37. Ako na zber a spracovanie požiadaviek?

38. Ako začíname s návrhom?

39. Zabezpečenie kvality vo fázach špecifikácie a skorého návrhu

Pomocou recenzie, kontroly, opravy.

40. Dokumentácia požiadaviek

41. Vývojový tím

42. Zákazník a jeho roly

43. Manažment financií

44. Manažment hardvéru a iných hmotných zdrojov

45. Manažment návrhu a implementácie

46. Zabezpečenie kvality vo fázach návrhu a implementácie

47. Dokumentácia návrhu a implementácie

48. Ako na zmeny a evolúciu?

49. Manažment testovania

50. Testovanie modulov

Overenie kvality integrity a zapuzdrenia modulov a kvality kódu.

51. Integračné testovanie

Overenie kvality rozhraní modulov a ich implementácií.

52. Testovanie systému

Overenie kvality prispôsobenia pracovnému prostrediu a kvality konfigurácií.

53. Akceptačné testovanie

Overenie kvality systému ako celku, validácia a určenie jeho úžitkovej hodnoty.

54. Využitie požiadaviek pri testovaní

55. Zabezpečenie kvality vo fáze testovania

56. Dokumentácia k testovaniu

57. Retest

Konformačný test. Spúšťanie testovacích prípadov, ktoré v minulosti zlyhali ale následkom zmien v kóde by už mali byť úspešne vykonané.

58. Regresný test

Testuje sa vplyv zmeneného kódu na predtým otestovaný kód (ktorý bol zväčša v poriadku pred zmenou).

59. Retrospektíva

60. Hodnotenie tímu a jednotlivcov

61. Ako funguje CMMI?

62. 5 úrovní CMMI

63. CMMI a agilný vývoj

64. CIM

65. TMM

66. ITIL V3

67. ITIL 4

68. Basic COCOMO

Basic COCOMO počíta úsilie pri vývoji softvéru (a náklady) ako funkciu veľkosti programu. COCOMO 81 sa vzťahuje na tri triedy softvérových projektov: Organic projects – „malé“ tímy s „dobrými“ pracovnými skúsenosťami s „menej ako rigidnými“ požiadavkami. Semi-detached projects – „stredné“ tímy so zmiešanými skúsenosťami práce so zmesou rigidných a menej rigidných požiadaviek. Embedded projects – vyvinuté v rámci súboru „pevných“ obmedzení. Je to tiež kombinácia organických a semi-detached projektov.

69. Intermediate COCOMO

Maje 15 atribútov, každý dostane hodnotenie od 1 do 6, ktorý sa pohybuje od „veľmi nízkeho“ po „mimoriadne vysoké“ (dôležitosťou alebo hodnotou). Na rating sa vzťahuje multiplikátor úsilia. Produkt všetkého úsilia multiplikátorov viedie k faktoru úpravy úsilia (EAF). Typické hodnoty pre EAF rozsah od 0,9 do 1,4.

Product attributes, Hardware attributes, Personnel attributes, Project attributes.

70. Detailed COCOMO

Detailed COCOMO zahŕňa všetky charakteristiky Intermediate verzii s hodnotením vplyvy na každom kroku (analýza, návrh atď.) inžinierskoho procesu softvéru.

Päť fáz Detailed COCOMO: plan and requirement, system design, detailed design, module code and test, integration and test.