Okruhy otázok podľa moodle

1. Kto meria úspešnosť projektu?

-metrika, softvérové meranie

2. Čo je metrika?

-je hocijaký nápad, ktorým si inšpirovaný pri pozeraní/čítaní textu (nie len textu)
-napr: riadky kódu, počet metód, komentáre, operátory a operandy, správanie objektu počas vykonávania

3. Dve triedy softvérových metrík

-produkty – programy, dokumentácia, reporty
-procesy – ktorými bol produkt vyvíjaný (životné cykly, fázy špecifikácie, dizajnu, implementácie, testovanie a údržba)

4. Čo je meranie?

-meranie je pozorovanie, ktoré je vykonávané veľmi pozorne. Používajú sa na identifikovanie nových javov
-testujú predpokladané javy
-každé meranie vyžaduje stupnicu alebo metriku
-Meranie je mapovanie z empirického sveta do formálneho, relačného sveta. V dôsledku toho je meranie číslo alebo symbol priradený objektu pomocou tohto mapovania aby určilo atribút

5. Matematická notácia pre metriky

-meranie je mapovanie z empirického sveta do formálneho, relačné sveta. V matematike je metrika funkcia - m: X-> R, kde X je ľubovoľný typ, R je skupina kladných racionálnych čísel, kde pre všetky x patriace do X platí:

- m(x) je definované
- m(x) >= 0
- m(x1 U x2) <= m(x) + m(x2)

6. Externé metriky produktu

-vychádza z klasifikácie podľa I.Sommervilla, ktorý rozdeľuje metriku na metriku produktov (merajú vlastnosti softvérových produktov) a metriku procesov (merajú vlastnosti procesov použitých na získanie týchto produktov)
-metrika produktu ďalej zahrňuje 2 kategórie
    -externá metrika produktu – vlastnosti viditeľné používateľovi
    -interná metrika produktu – vlastnosti viditeľné tímu vývojárov
-metrika nespoľahlivosti – určuje počet zostávajúcich chýb
-metrika funkcionality – posudzuje akú užitočnú funkcionalitu produkt poskytuje (AMEISE)
-metrika výkonu – posudzuje produktom využívané zdroje ako rýchlosť výpočtu, priestoru a obsadenosti

-metrika použiteľnosti – posudzuje aká je jednoduchosť učenia a používania produktu
-metrika ceny – posudzuje náklady na obstarávanie a používanie produktu

7. Interné metriky produktu

-metrika veľkosti – poskytuje merania aký veľký produkt je interne
-metrika zložitosti - súvisí s metrikou veľkosti a posudzuje aký zložitý je produkt
-metrika štýlu – posudzuje dodržiavanie písania návodov pre komponenty produktu ako sú programy a dokumenty

8. Metriky projektu

-hm, zber dát a vykazovanie hodnot metriky v pravidelných intervaloch
-vytvorenie konkrétnych cieľov zlepšenia

9. LOC

-Lines of Code – riadky kódu
-počíta sa zväčša pre spustiteľné príkazy
-môže sa to zamieňať, viď program v Assembleri kde LOC a príkaz inštrukcií je to isté
-rozlišujeme 2 typy LOC, example -- for(i=0; i<100; i++) printf(„hello“);
- LOC-phy, teda fyzické riadky kódu, pre example to je 1
- LOC-log, teda logické riadky kódu, pre example to je 2

10. Cyklomatická zložitosť (všeobecne)

-navrhnutá McCabe v roku 1976
-testovateľnosť a zrozumiteľnosť programov
-počet lineárne nezávislých ciest, ktoré zahŕňa program, inými slovami povedané je to údaj ktorý udáva počet cyklov, ktorý sa vykoná v programe počas testovania programu
-určuje úsilie požadované na testovanie programu
-kontrolné štruktúry – sekvencie, vetvenie, cykly
-vzorec V(G) = e – n + 2p, (Cyklomatické číslo G) , kde e – počet edges (čiar), n – počet uzlov, p – počet pripojených komponentov

11. Cyklomatická zložitosť sekvencie

-pre jednoduchú sekvenciu je v = 1-2+2=1

12. Cyklomatická zložitosť vetvenia

-pre jednoduché if then else vetvenie je v = 4-4+2=2

13. Cyklomatická zložitosť cyklu

-pre jednoduchý while cyklus je v = 3-3+2=2

14. Fan-in a Fan-out

-najbežnejšia dizajnová štruktúra metriky (tvorcovia Constantine a ešte Myers)

-Fan-in nám udáva počet modulov, ktoré volajú daný modul
-Fan-out nám udáva počet modulov, ktoré sú volané daným modulom
-vo všeobecnosti, moduly s veľkým fan-in sú relatívne malé a jednoduché a zvyčajne uložené na spodnej vrstve dizajnovej štruktúry. Naopak moduly ktoré sú veľké a komplexné majú zväčša malý fan-in
-z toho vyplýva, že moduly s veľkým fan-in a veľkým fan-out sú biedne navrhnuté a je potreba ich predizajnova

15. FPA - základný princíp

-FPA – function point analysis
-FP metrika bola z väčšej časti vyvinutá aby pomohla spravodlivo určovať ceny za vývoj softvéru. Počet FP je rozdielny ako úsilie potrebné na vývoj systému
-vplyv FPA na plánovanie projektu – určovanie rozsahu, posúdenie dopadu náhrady, ceny náhrady, požiadavky vyhodnotenia, prideľovanie testovacích prostriedkov, posúdenie rizík, monitorovanie progresu
-môže byť aplikovaná v ktorejkoľvek fáze vývoja, zvyčajne ale na odhad ceny
-využívaná sa preto lebo funkcionalita nemôže byť meraná

16. FPA - ako na to?

-ako použiť FPA v 5tich krokoch
-musíme vypočítať celkový počet, ktorý bude použitý na definovanie zložitosti projektu (UFP – Unadjusted funcion points)
-musíme zistiť zložitosť nastavenia hodnôt (VAF – Value adjustment factor, AFP = UFP*VAF)
-musíme zistiť LOC, vybraním programovacieho jazyka, ktorý použijeme
-posledným krokom je vybranie zložitosti softvérového projektu z COCOMO tabuľky (Constructive Cost Model)
-no a teraz by sme mali vypočítať hodnoty pre effort (úsilie) a duration (trvanie)

17. COCOMO 81

-COCOMO – Constructive Cost Model, algoritmický softvér pre odhad nákladov modelu. Využíva základné regresné zložene s parametrami, ktoré sú odvodené od historických dát projektu ako aj súčasných a budúcich charakteristík projektu
-existujú 2 verzie: COCOMO 81 (waterfall - vodopád), COCOMO II (moderné SW procesy)
-*COCOMO 81*
-skladá sa z hierarchie troch podrobných a presných foriem, Organic, Semidetached, Embedded
-prvá úroveň Basic je dobrá na rýchly a hrubý odhad nákladov na softvér, presnosť je však obmedzená pre nedostatok faktorov
-druhá úroveň Intermediate – už berie do úvahy tieto faktory
-tretia úroveň Detailed – dodatočné vyúčtovanie pre vplyv jednotlivých fáz projektu

18. COCOMO II

-COCOMO – Constructive Cost Model, algoritmický softvér pre odhad nákladov modelu. Využíva základné regresné zložene s parametrami, ktoré sú odvodené od historických dát projektu ako aj súčasných a budúcich charakteristík projektu
-existujú 2 verzie: COCOMO 81 (waterfall - vodopád), COCOMO II (moderné SW procesy)
-*COCOMO II*
-tvorba investície alebo finančné rozhodnutia zahrňujúce úsilie pri vývoji softvéru
-rozpočet a rozvrh projektu ako základ pre plánovanie a riadenie
-určovanie ceny softvéru
-rozhodovanie, ktoré časti softvéru vyvíjať, opätovne použiť, prenajať alebo nakúpiť
-nastavenie zmiešaných investičných stratégií s cieľom vylepšiť softvér cez opätovné použitie, nástroje outsourcing
-rozhodovanie ako implementovať procesy na zlepšenie stratégie, ako je poskytované v SEI CMM

19. Code-and-fix model

-kódenie a fixovanie stále dokola.. to čo sa nakódi sa potom pofixuje atď atď

20. Vodopádový model

-špecifikácia a analýza, návrh, implementácia, testovanie, nasadenie systému, prevádzka

21. Vylepšený vodopádový model

-tak ešte lepši vodopádový model :D

22. UP

-Unified process - je to známy interaktívny developerský framework. Patrí tam OpenUp, AgileUp

23. Cleanroom

-je developerský proces slúžiaci na výrobu softvéru ktorý spĺňa určitý level spoľahlivosti

24. V-model

-je to predĺžený waterfall, procesy v obrázku časovo zľava doprava v tvare písmena V

25. Špirálový model

-je model pre softvérové projekty, založený na unikátnych riskantných vzorcoch

26. Scrum

-rozdeľuje prácu na určité stupne: vlastník, scrum master, tím ľudi

-má svoj životný cyklus, zväčša sa iteruje v 2-4 týždenných iteráciách, je tam nastavenie cieľov na začiatku, potom vývoj, na konci rekapitulácia čo sa stihlo a čo nie prečo atď..

-mnoho spoločností si ho osvojilo, je však len časťou toho čo je požadované pre doručenie sofistikovaného riešenia pre zúčastnené strany

-treba vypĺňať medzery, ktoré scrum ignoruje. Zas iné metódy sú rozličné čo môže byť neskôr mätúce pre obe strany

27. RAD

-dáva menší dôraz na plánovacie úlohy a väčší na vývoj

28. DSDM

-je projektový framework, hlavne použíteľný ako softvérová vývojová metóda

29. XP, párové programovanie

-XP (extreme – extrémne programovanie) – je agilná metodológia softvérového vývoju, ktorá predpisuje určité činnosti všetkým zúčastnením vývojového procesu. Činnosti sú ale vedené do extrému, vďaka tomu by sa malo toto programovanie lepšie prispôsobiť sa zmenám požiadavkám zákazníka a dodať softvér vo vyššej kvalite

30. TDD

-Test Driven Development je založený na malých stále sa opakujúcich krokoch vedúcich k lepšej efektivite celého systému. Jednoducho povedané najprv sa definuje funkcionalita, potom sa napíšu testy, ktoré túto funkcionalitu overujú a až tak príde na rad napísanie kódu a úprava toho kódu

31. FDD

-Feature Driven Development, základným modelom v tejto metodológii je doménový model na vysokej úrovni abstrakcie. Popisuje celý systém a slúži tak k minimalizácii problémov a spolupráci jednotlivých častí vytvorené rôznymi programátormi
-dá sa preložiť ako vývoj riadený užitočnými vlastnosťami.. nah

32. Prototypovanie

-tak dačo vykecať už z PRPS, nejaké tools, nejaké postupy, cez doménovú analýzu, konceptuálny model, scenáre, persony, protyp..

33. DAD

-Disciplined Agile Delivery
-poskytuje viac súdržný prístup k dosiahnutie doručenia agilného riešenia
-umožňuje zjednodušiť procesné rozhodovanie pri vývoji a správe softvéru; stavia na mnoho postupoch agilného softvérového vývoja viď Scrum, agilné modelovanie..
-rozvíja scrum pomocou agilného modelovania, XP,UP,Kanban..
-bol navrhnutý v IBM ale voľne dostupný a nevyžaduje žiadne IBM tools; je riadený cieľmi, škálovateľný a zohľadňuje podnik
-má životný cyklus založený na porovnávaní rizík a prínosov

-životný cyklus:
-zahŕňa rady o technický postupoch ako tie z XP rovnako ako modelovanie, dokumentácia
-pomáha prijať stratégie, ktoré sú pre Vás tie pravé
-poskytuje poradenstvo týkajúce sa realizovateľných alternatív a to popisovaním čo pracuje čo nepracuje a prečo
-rozširuje konštrukčne zameraný životný cyklus scrumu
-zameriava sa na doručenie spotrebného riešenia nie len na softvér na odoslanie

34. Povedomie (agilného) tímu
35. UML

-pretože vizualizácie sú lepšie ako text
-UML – Unified Modeling Language
-graficko-vizuálny jazyk, diagramy, veľa nástrojov na kreslenie, plne definovaná syntax, konkrétne domény, editovateľná syntax a sémantika prostredníctvom UML profilov
-pomocou UML diagramov potom vieme vytvoriť ďalšie ako sú use case diagram, class diagram, activity diagram..

36. AMEISE

-architektúra klient – server, kde server je simulačný prostriedok, a klient je v interakcii so simulačným prostriedkom
-Aorta – dalši klient pre admina
-je to dedičný systém
-real job to manage, vraj to čo sme skúšali na cviku cez Ameise bolo kódenie Maisu:))

37. Manažment špecifikácie a skorého návrhu

-analýza, špecifikácia, systémový dizajn, kvalita, dokumentácia a vývoj, zdroje: ľudia, softvér, hardvér, peniaze
-analýza domény a problémy – inžinierstvo domén, inžinierstvo systému, popis problému, požiadavky, odborníci domény

38. Ako na zber a spracovanie požiadaviek?

-komunikácia so zákazníkom, zisťovanie presných požiadaviek, FP analýza, odhad ceny, odhad zdrojov
-najlepšie praktiky – rozhovory, bezkontextové otázky, dotazníky, pozorovanie, analýza dokumentov, analýza existujúcich systémov, prototypovanie, účel a cieľ analýzy

39. Ako začíname s návrhom?

-vždy je návrh založený na základe požiadaviek
-prototypovanie špecifických častí, ktoré pomáhajú s RE
-vytváranie architektúry systému založenej na potrebách zákazníka
-návrh manuálov a rozhraní
-príprava pre servis a údržbu

40. Zabezpečenie kvality vo fázach špecifikácie a skorého návrhu

-tri aspekty – review (posudok), inspection (inšpekcia), correction (oprava)
-inšpekcia – kontrola procesu, či je všetko vykonané tak ako by malo byť, zahŕňa posudok finálnych a pracovných verzií dokumentov
-posudok – kontroluje výstupy z procesov, z finálnych dokumentov
-oprava – zmena v procese alebo v dokumente

41. Dokumentácia požiadaviek

-príučky pomocou špeciálnej šamblóny
-požiadavky špecifikácie: CRC karty, formálne metódy, voľný text, tabuľky, UML diagramy, User stories (kto a čo v systéme má robiť)
-sú menej dôležité pri agilnom vývoji

42. Vývojový tím

-Tím – skupina ľudí, spolupráca, organizovanie tímu, motivácia
-Zákazník – podľa jeho chuti
-Soft skills – komunikácia, schopnosť pracovať skupine, osobnostné skills ako organizátor, líder, detektív, konzultant..

43. Zákazník a jeho roly
44. Manažment financií

-platy, čas zákazníka, náklad na SW/HW, poplatky

45. Manažment hardvéru a iných hmotných zdrojov

-administrácia, vývoj, skladovanie (databáza napr), servis a údržba

46. Manažment návrhu a implementácie

-správa vývoja
-systémový návrh, návrh modulov, kódenie, integrácia
-kvalita, dokumentácia a vývoj, zdroje: ľudia, softvér, hardvér, peniaze

47. Zabezpečenie kvality vo fázach návrhu a implementácie

-verification, overovanie – robíme to dobre?
-inspection (inšpekcia) – kontrolovanie procesov
-review(posudok) – kontrolovanie výstupov procesov
-simulation(simulácia) - testovanie programátorom, simulovanie operácii v inom prostredí
-correction(opravy) – zmeny v procese alebo dokumente
-documentation(dokumentácia) – vstupy pre testovanie

48. Dokumentácia návrhu a implementácie

-príručky podľa špecifických šablón
-vizuálne a textové dokumentácie: CRC karty, formálne metódy, tabuľky, uml diagramy, zdrojové kódy

-menej dôležité v agilnom vývoji

49. Ako na zmeny a evolúciu?

-opravy, vývoj na základe sebahodnotenia a zákazníckej spätnej väzby
-vydávať pravidelné verzie

50. Manažment testovania

-testovanie modulov, integračné testovanie, testovanie systému, akceptačné testovanie
-zlepšuje kvalitu, dokumentácia a vývoj, zdroje: človek, softvér, hardvér, peniaze

51. Testovanie modulov

-samostatné testy pre každý podsystém
-viac komunikácie/interakcie a komplexné rozhrania
-zameranie na - jednotka programu
-písanie správ o testovaní modulov
-kontrola limitov funkčnosti modulov a prevádzky

52. Integračné testovanie

-žiadne známky (mocks – náčrty abo dačo také) pre moduly
-systémové volania by mohli zatlačiť
-testovanie databázy
-zameranie na - interakcia modulov
-overenie a validácia modulov/systémového návrhu
-písanie záznamov integračného testovania

53. Testovanie systému

-žiadne známky pre moduly
-žiadne zatláčanie systémových volaní
-testovanie databázy
-test interakcie s prostredím
-zameranie na - systém vo vlastnom prostredí
-overovanie návrhu systému
-písanie reportov o testovaní systému

54. Akceptačné testovanie

-reálne údaje v databáze
-testovanie interakcie používateľa
-zameranie hlavne na - používateľa
-overovanie systému
-písanie hlásenia o použiteľnosti a požiadavkách o zmenu

55. Využitie požiadaviek pri testovaní

-tak je tam taký graf v prednáškach ale kto vie na čo sa snaží poukázať, asi že požiadavky by sa mali potom otestovať abo nevieeem uš

56. Zabezpečenie kvality vo fáze testovania

-testovanie modulov – kvalita modulov a kódu
-integračné testovanie – kvalita rozhraní a ich implementácia
-testovanie systému – kvalita adaptácie na prostredie, kvalita konfigurácie
-akceptačné testovanie – kvalita systému, obchodná hodnota, odhad a validácia

57. Dokumentácia k testovaniu

-Reporty využívajúce špecifické šablóny
-vizuálna a textová dokumentácia – vložíme sufix alebo prefix TEST
-skúšobná hodnotiaca správa
-dôležité v agilnom vývoji

58. Retest

-testovanie pri ktorom beží prípad testu, ktorý zlyhal keď bol spustený aby overil úspešnosť nápravy

59. Regresný test

-testovanie skôr testovaných programov, aby zabezpečil, že vady neboli zavedené v nezmenených oblastiach, ako výsledok iných zmien. Vykonáva sa keď je zmenený softvér alebo prostredie

60. Retrospektíva

-štruktúrovaný spôsob ako zachytiť získané poznatky a vytvoriť špecifický plán v krokoch pre zlepšenie ďalších projektov alebo fáz v projekte
-retrospektívne stretnutie: stretnutie na konci projektu, počas ktorého členovia tímu vyhodnocujú projekt, lekcie, ktoré sa naučili a môžu byť použité v ďalšom projekte

61. Hodnotenie tímu a jednotlivcov

-osobný a profesijný rast
-Informovanosť tímu (povedomie tímu)
-teamwork – tímová spolupráca

62. Ako funguje CMMI?

-CMMI – Capability Maturity Model Integration – je model základných procesov v organizácií spolu s odporúčaním ako ich efektívne implementovať. Zahŕňa v sebe tri oddelené modely:
    -CMMI for development (CMMI-DEV) – vývoj produktov a služieb
    -CMMI for Services (CMMI-SVC) – služobné ustanovenia, manažment
    -CMMI for Acquisition(CMMI-ACQ) – prínos pre produkty a služby
-CMMI neponúka organizáciam certifikáty ale môžu byť organizácie ocenené

-snažia sa o zlepšenie životného cyklu produktu, implementovanie ISO štandardov, integrovať skúsenosti z iných oblastí, tesnejšie previazať inžinierske a riadiace aktivity

63. 5 úrovní CMMI

-Initial (Počiatočný) – počiatočný bod pre použitie nových alebo nezdokumentovateľných opakovaných postupov
-Repeatable(Opakovateľný) – tento proces je aspoň dostatočne zdokumentovaný, takže môžeme skúsiť opakovať rovnaké kroky
-Defined(Definovaný) – proces je definovaný/potvrdený ako štandard
-Managed(Riadený) – proces je riadený kvantitatívne v súlade s dohodnutými metrikami
-Optimizing(Optimalizácia) – riadene procesov zahŕňa premyslený proces optimalizácie/zlepšenia

64. CMMI a agilný vývoj

-nie je to nemožné
-DAD, SCRUM, FDD atď.. toto všetko je OK
-XP – je to procesné riadenie požiadaviek, nie je možné zlúčiť s CMMI

65. CIM

-CIM – Capability Immaturity Model
-model od Finkelsteina (rozdeľuje na pochabý, hlupák, blázon) a Schorscha (rozdeľuje na nedbalý, obštrukčný, pohŕdavý, podkopávač)
-je to len akousi paródiou, resp. poskytnúť kontrast k CMM (Capability Maturity Model)

66. TMM

-Level1 – initial(počiatočný) – používanie ad hoc metód (ad hoc – bezdrôtové testovanie..) pre testovanie, preto výsledok nie je opakovateľný a nemá žiaden štandard kvality
-Level2 – defenition(definovanie) – pri testovaní sa používajú stratégie, plány testovania, prípady testovania založené na požiadavkách. Testovanie nezačína pokiaľ nie je projekt hotový. Testovanie porovnáva produkt s požiadavkami
-Level3 – integration(integrácia) – testovanie je vložené do softvérového životného cyklu
-Level4 – management and measurement(manažment a meranie) – je obsiahnuté vo všetkých fázach životného cyklu softvéru, zahrňujúce reviews, dizajn, požiadavky..
-Level5 – optimization(optimalizácia) – samotný testovací proces je testovaný a vylepšovaný v každej iterácií

Otázky zo screenshotov, resp. pravdepodobne z testov na modli

Čo je verifikácia?

-proces na určenie toho, či vytvárame produkt správne

Na čo slúžia testy modulov?

-Overenie kvality integrity a zapúzdrenia modulov a kvality kódu

Čo je retestovanie(retesting)?

-spúšťanie testovacích prípadov, ktoré v minulosti zlyhali ale následkom zmien v kóde by už mali byť úspešné

-konfirmačné testovanie

Ktorá postupnosť činností vystihuje vodopádový model vývoja programu?

-špecifikácia, návrh, implementácia, testovanie

Na čo slúžia akceptačné testy?

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

Ako definujeme chuť/túžbu zákazníka?

-FP(m+1) ~ FP(m)*1,01

Charakterizujte testovanie softvéru

-testovanie je spúšťanie programu za účelom nájdenia chýb

Čo je skratka COCOMO

-Constructive Cost Model

Aký bude odhad počtu vývojárov na projekte o veľkosti 1500 funkčných bodov na základe experimentálne overených pravidiel?

-10

Aké sú výhody zákazníka?

-nepoberá od nás výplatu

-je jediný, kto vie odhaliť určité chyby

Čo je cieľom statickej analýzy kódu?

-odhaliť inštrukcie, ktoré nikdy nebudú vykonané

-nájsť prázdne cykly

-nájsť redundatné deklarácie

Čo je validácia?

-proces na určenie toho, či vytvárame ten správny produkt

Aký bude odhad počtu ľudí zodpovedných za údržbu na projekte o veľkosti 1500 funkčných bodov na základe experimentálne overených pravidiel?

-3

Čo je korekcia?

-oprava chýb vo výsledkoch procesov

Čo je regresné testovanie?

-Testuje sa vplyv zmeneného kódu na predtým otestovaný kód (ktorý bol zväčša v poriadku

pred zmenou)

Čo je inšpekcia?

-kontrola priebehu procesov

Za akým účelom sa zaviedol pojem FP – function point

-hodnota FP udáva náročnosť vytvorenia aplikácie. Stanovením tejto hodnoty sa dajú odhadnúť personálne a finančné nároky na tvorbu programu