

T1

Softvérové inžinierstvo

Definícia:

- Softvérové inžinierstvo je inžinierska disciplína, ktorá sa zaoberá teóriami, metódami a nástrojmi pre profesionálny vývoj softvéru.
- Zameriava sa na nákladovo efektívny vývoj softvéru.
- Zahŕňa všetky aspekty výroby softvéru, od počiatočnej špecifikácie systému až po údržbu systému po jeho uvedení do prevádzky.

Základné činnosti:

1. **Špecifikácia softvéru:** Definovanie softvérových požiadaviek v spolupráci so zákazníkmi a inžiniermi.
2. **Vývoj softvéru:** Návrh a programovanie softvéru.
3. **Validácia softvéru:** Zabezpečenie, aby softvér spĺňal požiadavky zákazníka.
4. **Evolúcia softvéru:** Prispôsobovanie softvéru meniacim sa požiadavkám zákazníkov a trhu.

Inžinierska disciplína:

- Zahrňuje aplikáciu vhodných teórií a metód na riešenie problémov pri zohľadnení organizačných a finančných obmedzení.

Softvér

- Označuje počítačové programy a súvisiacu dokumentáciu.
- Softvérové produkty môžu byť vyvinuté pre konkrétnych zákazníkov alebo pre všeobecný trh.

Softvérové Produkty

Generické Produkty

- Samostatné systémy predávané každému zákazníkovi, ktorý si ich želá kúpiť.
- Špecifikáciu a rozhodnutia o zmene softvéru vlastní vývojári.

Prispôsobené Produkty

- Softvér objednaný konkrétnym zákazníkom na splnenie ich potrieb.
- Špecifikáciu a rozhodnutia o zmene softvéru vlastní zákazníci.

Spoľahlivosť Systému

- Spoľahlivosť systému odzrkadľuje dôveru používateľov v systém.

Časté Príčiny Porúch Prevádzky:

- Chyby ľudských operátorov, často hlavná príčina zlyhaní v sociálno-technických systémoch.

Kľúčové Charakteristiky Spoľahlivého Systému:

1. **Dostupnosť:** Pravdepodobnosť, že systém bude v prevádzke a schopný poskytovať užitočné služby používateľom.
2. **Výpočtová Spoľahlivosť:** Pravdepodobnosť, že systém bude správne poskytovať služby podľa očakávania používateľov.
3. **Ochrana Zdravia, Života a Prostredia:** Posúdenie pravdepodobnosti, že systém môže spôsobiť škody ľuďom alebo životnému prostrediu.
4. **Informačná Bezpečnosť:** Hodnotenie schopnosti systému odolávať náhodným alebo úmyselným prienikom (C-I-A: dôvernosť, integrita, dostupnosť).
5. **Odolnosť:** Hodnotenie, ako dobre systém udrží kontinuitu kritických služieb v prítomnosti rušivých udalostí, ako je zlyhanie zariadení a kybernetické útoky.

Typy Kritických Systémov:

Systémy môžu byť kritické z hľadiska:

- bezpečnosti (napr. ochrana rastlín v chemických systémoch)

- misie-kritické (napr. navigačné systémy vesmírnych lodí)
- obchodné-kritické (napr. systémy účtovníctva v banke).

Náklady na Spoľahlivosť

Náklady na spoľahlivosť majú tendenciu exponenciálne rásť, pretože je potrebná vyššia úroveň spoľahlivosti

Dva hlavné dôvody:

1. **Použitie Drahých Vývojových Techník a Hardvéru:** Potrebné na dosiahnutie vyšších úrovní spoľahlivosti.
2. **Rozšírené Testovanie a Overovanie:** Potrebné na presvedčenie klientov a regulátorov systému, že boli dosiahnuté požadované úrovne spoľahlivosti.

T2

Softvérový proces:

- **Špecifikácia:** Definovanie toho, čo má systém robiť.
- **Návrh a implementácia:** Definovanie organizácie systému a implementácia systému.
- **Validácia:** Kontrola, či systém vyhovuje požiadavkám zákazníka.
- **Evolúcia:** Zmena systému v reakcii na meniace sa potreby zákazníkov.

Plánom riadené a agilné procesy:

- **Procesy riadené plánom:** Všetky procesné činnosti sú vopred naplánované a pokrok sa meria podľa tohto plánu.
- **Agilné procesy:** Plánovanie je prírastkové a umožňuje jednoduchšie zmeny procesu podľa meniacich sa požiadaviek zákazníkov.

Softvérové procesné modely:

- **Vodopádový model:** Oddelené fázy špecifikácie a vývoja.
- **Postupný vývoj:** Špecifikácia, vývoj a validácia sú navzájom prepojené.
- **Integrácia a konfigurácia:** Systém je zostavený z existujúcich konfigurovateľných komponentov.

(1) Vodopádový model:

- **Fázy:** Analýza a definícia požiadaviek, Návrh systému a softvéru, Implementácia a testovanie jednotiek, Integrácia a testovanie systému, Prevádzka a údržba.
- **Problémy:** Neflexibilné rozdelenie projektu, vhodný len pre dobre pochopené požiadavky.

(2) Postupný vývoj:

- **Výhody:** Znížené náklady na prispôsobenie sa zmenám, rýchlejšie dodanie a nasadenie.
- **Problémy:** Nedostatok viditeľnosti, degradácia štruktúry systému.

(3) Integrácia a konfigurácia:

- **Založené na opätovnom použití softvéru.**
- **Výhody a nevýhody:** Znížené náklady a riziká, ale strata kontroly nad vývojom opätovne použitých prvkov systému.

Procesné činnosti:

(1) Špecifikácia softvéru:

- Získavanie a analýza požiadaviek.
- Špecifikácia a overenie požiadaviek.

(2) Návrh a implementácia softvéru:

- Návrh softvéru.
- Implementácia softvéru.

(2a) Dizajnérske činnosti:

- Architektonický návrh.
- Návrh databázy.
- Návrh rozhrania.
- Výber a dizajn komponentov.

(2b) Implementácia systému:

- Programovanie.
- Ladenie.

(3) Validácia softvéru:

- Verifikácia a validácia (V & V) pre kontrolu a hodnotenie systému.
- Testovanie systému.

(4) Evolúcia alebo ďalší vývoj softvéru

Zníženie nákladov na prepracovanie:

- Predvídanie zmien.
- Tolerancia zmien.

Vyrovnanie sa s meniacimi sa požiadavkami:

- Systémové prototypovanie.
- Inkrementálne doručovanie.

Prototypovanie softvéru:

- Výhody: Vylepšená použiteľnosť, presnejšie prispôsobenie potrebám a kvalite dizajnu.
- Môže zahŕňať vynechanie funkcií.

Postupný vývoj a doručovanie:

- Postupný vývoj a prírastkové doručovanie s výhodami v podobe rýchlejšieho dodania a menšieho rizika.

Problémy s postupným doručovaním:

- Základy, iteratívne procesy a modely obstarávania.

T3

Projektový manažment

- Softvér je nehmotný, nie je viditeľný ani hmatateľný.
- Veľké softvérové projekty sú často jedinečné a ťažko predvídateľné.

Kapitola 23: Plánovanie projektu

- Plánovanie projektu zahŕňa delenie práce na úlohy a pridelenie členov tímu.
- Projektový plán pomáha informovať tím a zákazníkov a hodnotiť pokrok projektu.

Univerzálne manažérske činnosti

- Plánovanie projektu: Odhadovanie a pridelenie úloh.
- Riadenie rizík: Hodnotenie a riešenie rizík projektu.
- Riadenie ľudí: Výber tímov a efektívna tímová spolupráca.

Plánovanie projektu

- Rozdelenie práce na úlohy a predvídanie problémov.
- Projektový plán informuje tím a zákazníkov o postupe a pokroku.

Etapy plánovania

- Fáza návrhu: Uchádzanie sa o zákazku.
- Fáza spustenia projektu: Plánovanie tímu a rozdelenie úloh.
- Pravidelné úpravy počas projektu: Prispôsobenie plánu na základe skúseností.

Plánom riadený vývoj

- Podrobný prístup k riadeniu inžinierskych projektov.
- Obsahuje plán projektu, zodpovedné osoby, harmonogram a pracovné výstupy.

Projektové plány

- Určujú zdroje, časový harmonogram a čo sa má vykonať.

Prezentácia harmonogramu

- Stĺpcové grafy ukazujú aktivity a závislosti úloh.

Etapy agilného plánovania

- Plánovanie vydania: Rozhodovanie o funkciách na niekoľko mesiacov dopredu.
- Iteračné plánovanie: Plánovanie krátkodobých prírastkov systému.

Prístupy k agilnému plánovaniu

- Plánovanie v Scrum: Kontrola denného pokroku a riešenie problémov.
- Plánovacia hra: Meranie pokroku pomocou príbehov používateľov.

Klasifikácia rizika

- Riziká projektu: Ovládajú plán alebo zdroje.
- Riziká produktu: Ovládajú kvalitu alebo výkon softvéru.
- Podnikateľské riziká: Ovládajú organizáciu.

Proces riadenia rizík

- Identifikácia rizík: Rozpoznať riziká projektu, produktu a obchodné riziká.
- Analýza rizík: Posúdiť pravdepodobnosť a dôsledky rizík.
- Plánovanie rizika: Vypracovať plány na minimalizáciu rizika.
- Monitorovanie rizík: Sledovať riziká počas projektu.

Plánovanie rizik

- Stratégie vyhýbania sa: Minimalizácia pravdepodobnosti rizika.
- Stratégie minimalizácie: Minimalizácia vplyvu rizika na projekt alebo produkt.
- Pohotovostné plány: Plány na riešenie rizík, keď sa vyskytnú.

Typy osobností

- Orientovaný na úlohy: Motivovaný prácou samotnou.
- Sebaorientovaný: Práca ako nástroj na dosiahnutie individuálnych cieľov.
- Orientovaný na interakciu: Motivovaný prítomnosťou a činmi kolegov.

Tímová práca

Skupinová súdržnosť

- Výhody súdržnej skupiny:
 - Normy kvality vytvárajú členovia skupiny.
 - Učenie a znižovanie zábran.
 - Zdieľanie vedomostí

Efektívnosť tímu

- Rôznorodosť úloh v projektovej skupine.
- Organizácia podľa schopností.
- Dôležitá komunikácia medzi členmi a tímom softvérového inžinierstva.

Zloženie skupiny

- Problémy s rovnakou motiváciou:
 - Orientácia na úlohy.
 - Orientácia na seba.
 - Orientované na interakciu.

Skupinová organizácia

- Organizácia: malé skupiny neformálne, veľké hierarchická štruktúra.
- Agilný vývoj: neformálna skupina pre lepšiu informačnú výmenu.

T4

1. Inžinierstvo s požiadavkami

- Inžinierstvo požiadaviek zahŕňa tvorbu služieb a obmedzení systému podľa zákazníkových požiadaviek.

2. Čo je to požiadavka?

- Požiadavka môže byť abstraktným vyjadrením služby a má dvojitý účel: ponuka na zákazku a zmluva.

3. Druhy požiadaviek

- Požiadavky používateľov: Príkazy a diagramy pre zákazníkov.
- Požiadavky na systém: Štruktúrovaný dokument s podrobným popisom funkcií.

4. Funkcionálne a nefunkcionálne požiadavky

- Funkcionálne požiadavky popisujú, čo systém robí a ako reaguje.
- Nefunkcionálne požiadavky obmedzujú služby a môžu sa vzťahovať na časové obmedzenia a štandardy.

5. Doménové požiadavky

- Týkajú sa služieb a obmedzení systému v prevádzke.

6. Požiadavky na úplnosť a konzistentnosť

- Požiadavky by mali byť úplné a konzistentné bez rozporov.

7. Klasifikácia nefunkcionálnych požiadaviek

- Požiadavky na produkt určujú správanie dodaného produktu.
- Organizačné požiadavky vychádzajú z organizačných politík.
- Externé požiadavky sú dôsledkom externých faktorov.

8. Ciele a požiadavky

- Cieľ je všeobecný zámer používateľa.
- Overiteľná nefunkcionálna požiadavka je merateľný výrok.

Metriky na špecifikovanie nefunkcionálnych požiadaviek

Cieľ	Miera
Rýchlosť	<ul style="list-style-type: none">Spracované transakcie za sekunduČas odozvy používateľa/udalostiČas obnovenia obrazovky
Veľkosť	<ul style="list-style-type: none">Veľkosť v MBPočet čipov ROM
Jednoduchosť použitia	<ul style="list-style-type: none">Čas na tréningPočet pomocných rámcov
Spôľahlivosť	<ul style="list-style-type: none">Priemerný čas do zlyhaniaPravdepodobnosť nedostupnostiMiera výskytu porúchDostupnosť
Robustnosť	<ul style="list-style-type: none">Čas na reštart po zlyhaníPercento udalostí, ktoré spôsobili zlyhaniePravdepodobnosť poškodenia údajov pri zlyhaní
Prenosnosť	<ul style="list-style-type: none">Percento cieľových závislostí

Inžiniersky proces na špecifikáciu požiadaviek:

- Procesy na špecifikáciu požiadaviek sa líšia podľa domény, zainteresovaných ľudí a organizácií, ale existujú všeobecné činnosti:
 - Získavanie požiadaviek.
 - Analýza požiadaviek.
 - Validácia požiadaviek.
 - Riadenie požiadaviek.

Požiadavky a dizajn:

- Požiadavky by mali definovať, čo systém má robiť a aký má byť, zatiaľ čo dizajn popisuje, ako to systém má robiť.

Kontrolné kontroly:

- Overiteľnosť: Môže byť požiadavka testovaná?
- Zrozumiteľnosť: Je požiadavka správne pochopená?
- Vysledovateľnosť: Je jasne uvedený pôvod požiadavky?
- Prispôsobivosť: Dá sa požiadavka zmeniť bez veľkého vplyvu na ostatné požiadavky?

Plánovanie manažmentu požiadaviek:

- Rozhodnutia manažmentu požiadaviek zahŕňajú:

- Identifikáciu požiadaviek, aby boli jednoznačne identifikované.
- Proces riadenia zmien na vyhodnotenie vplyvu a nákladov zmien.
- Politiky sledovateľnosti na definovanie vzťahov medzi požiadavkami a návrhom systému.
- Podpora nástrojov, od systémov na správu požiadaviek po jednoduché databázové systémy.

Riadenie zmeny požiadaviek:

- Rozhodovanie o akceptovaní zmien požiadaviek zahŕňa:
 - Analýzu problému a špecifikáciu zmien.
 - Analýzu zmien a kalkuláciu účinku navrhovanej zmeny.
 - Implementáciu zmien v dokumentoch a systéme, ak je to potrebné.

T5

Modelovanie systému

1. *Systémové modelovanie* je proces vytvárania abstraktných modelov systému, pričom každý model predstavuje iný pohľad na daný systém.
2. *UML* (Unified Modeling Language) je vizuálny grafický jazyk, vyvinutý Boochom, Rumbaughom a Jacobsonom. UML slúži k vizualizácii, špecifikácii, konštrukcii a dokumentácii softvérových systémov. Podľa definície Object Management Group (OMG) je UML štandardizovaným spôsobom zápisu plánov systému, ktorý zahŕňa koncepčné prvky a konkrétne prvky.

UML

- Unified (Booch , Rumbaugh , Jacobson)
- Modeling (vizuálny, grafický)
- Language (gramatika, syntax, sémantika)

Systémové perspektívy:

- *Externá perspektíva*: Modeluje kontext alebo prostredie systému z vonkajšieho hľadiska.
- *Interakčná perspektíva*: Modeluje interakcie medzi systémom a jeho prostredím.
- *Štrukturálna perspektíva*: Modeluje organizáciu systému a štruktúru údajov, ktoré systém spracováva.
- *Perspektíva správania*: Modeluje dynamické správanie systému a jeho reakcie na udalosti.

Typy diagramov UML:

- *Diagramy aktivít*: Zobrazujú činnosti spojené s procesmi a spracovaním údajov.
- *Diagramy prípadov použitia*: Zobrazujú interakcie medzi systémom a jeho používateľmi.
- *Sekvenčné diagramy*: Zobrazujú interakcie medzi aktérmi a systémom.

- *Diagramy tried:* Zobrazujú triedy objektov v systéme a vzťahy medzi nimi.
- *Stavové diagramy:* Ukazujú, ako systém reaguje na udalosti.

Použitie grafických modelov:

- Uľahčujú diskusiu o existujúcom alebo navrhovanom systéme, aj keď nie sú úplné.
- Slúžia ako dokumentácia existujúceho systému, aj keď nemusia byť úplné.
- Môžu byť použité na podrobný popis systému a generovanie implementácie, ak sú správne a úplné.

(1) Externá perspektíva - Kontextové modely

- Kontextové modely sa používajú na ilustráciu operačného kontextu systému a ukazujú, čo sa nachádza mimo jeho hraníc.
- Architektonické modely ukazujú systém a jeho vzťahy s ostatnými systémami.

(2) Procesná perspektíva

- Procesné modely odhaľujú, ako sa vyvíjaný systém používa v širších obchodných procesoch.
- Na definovanie sa môžu použiť diagramy aktivít UML, ako sú modely podnikových procesov (workflow) a modely transformácií údajov (dataflow).

(3) Interakčné modely

- Modelovanie interakcie používateľov je dôležité pre identifikáciu požiadaviek používateľov.
- Modelovanie interakcie medzi systémami pomáha identifikovať komunikačné problémy.
- Modelovanie interakcie komponentov nám pomáha posúdiť, či navrhovaná štruktúra systému pravdepodobne poskytne požadovaný výkon a spoľahlivosť.
- Na modelovanie interakcií sa používajú UML diagramy prípadov použitia a diagramy sekvencií a komunikácie.

Sekvenčné diagramy

- Sekvenčné diagramy sú súčasťou UML a používajú sa na modelovanie interakcií medzi aktérmi a objektmi v rámci systému.

(4) Štrukturálne modely

- Štrukturálne modely softvéru zobrazujú organizáciu systému z hľadiska komponentov, ktoré tvoria tento systém a ich vzťahy.

Diagramy tried

- Diagramy tried sa používajú pri vývoji objektovo orientovaného modelu systému na zobrazenie tried v systéme a asociácií medzi týmito triedami.

(5) Modely správania

- Behaviorálne modely opisujú dynamické správanie systému pri jeho vykonávaní a ukazujú, čo sa deje pri reakcii systému na vstupy z okolia.
- Tieto stimuly sa rozdeľujú na údaje, ktoré musí systém spracovať, a udalosti, ktoré spúšťajú akcie v systéme.
- Modely stavových strojov zobrazujú stavy systému ako uzly a udalosti ako prechody medzi týmito stavmi. Stavové diagramy sú súčasťou UML a slúžia na reprezentáciu modelov stavových strojov.

Modelom riadené inžinierstvo

- Modelom riadené inžinierstvo (MDE) je prístup k vývoju softvéru, kde hlavnými výstupmi vývojového procesu sú skôr modely ako programy.

Použitie modelom riadeného inžinierstva

- Modelom riadené inžinierstvo je stále v ranom štádiu vývoja a nie je jasné, či bude mať významný vplyv na prax softvérového inžinierstva.
- Výhody zahŕňajú schopnosť posúdiť systémy na vyšších úrovniach abstrakcie a možnosť automatického generovania kódu, čo uľahčuje prispôbenie systému novým platformám.
- Nevýhody zahŕňajú to, že modely nie sú vždy vhodné na implementáciu a náklady na vývoj nových prekladačov pre nové platformy môžu vyvážiť úspory z generovania kódu.

Architektúra riadená modelom

- Architektúra riadená modelom (MDA) bola predchodcom obecného modelom riadeného inžinierstva.

T6

Architektonický dizajn

- Architektonický dizajn zaoberá organizáciou softvérového systému a jeho štruktúrou.
- Výstupom architektonického návrhu je model architektúry, ktorý opisuje, ako je systém zorganizovaný a komunikujú jeho komponenty.

Výhody explicitnej architektúry

- Komunikácia so zainteresovanými stranami:
 - Architektúra sa používa na diskusiu medzi zainteresovanými stranami systému.
- Systémová analýza:
 - Umožňuje analýzu, či systém môže splniť svoje nefunkčné požiadavky.
- Opätovné použitie vo veľkom meradle:
 - Architektúra môže byť opakovane použiteľná v rôznych systémoch.

Použitie architektonických modelov

- Uľahčuje diskusiu o návrhu systému:
 - Abstraktný pohľad na systém je užitočný pre komunikáciu a plánovanie projektov.
- Dokumentovanie architektúry:
 - Cieľom je vytvoriť model systému, ktorý zobrazuje komponenty, ich rozhrania a prepojenia.

Architektonické reprezentácie

- Krabicové a čiarové diagramy sú veľmi abstraktné, ale užitočné pre komunikáciu a plánovanie projektov.
- Niektorí tvrdia, že UML je vhodnou notáciou pre popis architektúr.
- Architektonické popisné jazyky (ADL) boli vyvinuté, ale nie sú široko používané.

Opätovné použitie architektúry

- Systémy v rovnakej doméne často majú podobné architektúry, ktoré odrážajú koncepty domény.
- Aplikačné produktové rady sú postavené na základnej architektúre s variantmi podľa požiadaviek zákazníkov.

Architektonické pohľady

- 4 + 1 pohľadový model softvérovej architektúry obsahuje:

- Logický pohľad (objekty alebo triedy objektov).
- Procesný pohľad (interagujúce procesy).
- Vývojový pohľad (rozloženie softvéru počas vývoja).
- Fyzický pohľad (hardvér systému a distribúcia softvérových komponentov medzi procesormi).
- Súvisiace prípady použitia alebo scenáre (+1).

Architektonické vzory

- Architektonický vzor je štylizovaný popis osvedčenej dizajnerskej praxe, ktorý bol overený v rôznych prostrediach.

(1) Organizácia systému

- Odráža základnú stratégiu štruktúrovania systému.
- Najčastejšie sa používajú tri organizačné štýly:
 - Štýl zdieľaného úložiska údajov
 - Štýl zdieľaných služieb a serverov
 - Abstraktný strojový alebo vrstvený štýl.

(1.1) Vrstvená architektúra

- Používa sa na modelovanie rozhrania podsystémov.
- Organizuje systém do vrstiev, pričom každá vrstva poskytuje rôzne služby.
- Podporuje postupný vývoj podsystémov v rôznych vrstvách.
- V niektorých prípadoch môže byť umelé.

(1.2) Zdieľané úložisko údajov

- Subsystémy zdieľajú údaje cez centrálnu databázu alebo vlastné distribuované úložiská.
- Model zdieľania úložiska sa používa na zdieľanie veľkého množstva údajov.

(1.3) Zdieľané služby a servery

- Model distribuovaného systému, ktorý zahrňuje údaje a spracovanie medzi rôznymi komponentmi.
- Môže byť implementovaný na jednom počítači.
- Obsahuje samostatné servery poskytujúce špecifické služby a klientov využívajúcich tieto služby.

(2) Modulárne štýly rozkladu

- Štýly rozkladu (pod)systémov na moduly.
- Modul je komponent systému, ktorý poskytuje služby iným komponentom, ale nepovažuje sa za samostatný systém.
- Dva modulárne modely rozkladu:
 - Objektový model, kde je systém rozložený na interagujúce objekty.
 - Model potrubia alebo toku údajov, kde je systém rozložený na funkčné moduly transformujúce vstupy na výstupy.

(2.2) Architektúra potrubia a filtra

- Varianty tohto prístupu sú časté, najmä v systémoch na spracovanie údajov.
- Dávkový sekvenčný model sa používa, keď sú transformácie postupné.

(3) Štýly ovládania

- Zaoberajú sa riadiacim tokom medzi podsystémami, na rozdiel od modelu rozkladu systému.

(3.1) Centralizované ovládanie:

- Jeden podsystém riadi ostatné.
- (3.1.1) Model odovzdania a vrátenia riadenia: Riadenie sa pohybuje zhora nadol, vhodné pre sekvenčné systémy.
- (3.1.2) Manažérsky model: Použiteľný pre súbežné systémy, kde jeden komponent riadi ostatné procesy.

(3.2) Ovládanie založené na udalostiach

- Udalosti sú externe generované a riadenie je založené na nich.

- Dva hlavné modely riadené udalosťami:
 - (3.2.1) Vysielací model: Udalosť je vysielaná do všetkých podsystémov, ktoré majú záujem.
 - (3.2.2) Modely riadené prerušením: Používa sa v systémoch v reálnom čase s rýchlym spracovaním udalostí.

(3.2.1) Vysielací model

- Efektívny pri integrácii podsystémov na rôznych počítačoch v sieti.
- Podsystémy registrujú záujem o konkrétne udalosti, ktoré sú im zasielané.
- Riadiaca politika nie je vložená do správy udalostí, ale rozhoduje o nich samotné podsystémy.

(3.2.2) Riadenie prerušením

- Používa sa v systémoch v reálnom čase, kde je rýchla reakcia nevyhnutná.
- Existujú typy prerušení s preddefinovaným spracovaním.
- Každý typ prerušenia je spojený s umiestnením pamäte a prepínač prerušení vedie k spracovaniu udalosti.
- Zabezpečuje rýchlu odozvu, ale vyžaduje zložité programovanie a overenie.

1. Čo je pre systém dôležitejšie a prečo – spoľahlivosť alebo výkonnosť (dependability vs. performance)?

Spôľahlivosť, lebo výkonný ale nespoľahlivý systém nebude akceptovaný/používaný

2. Aká je najčastejšia/hlavná príčina zlyhania systémov?

Chyba v hardvéri

Chyba v softvéri

Chyba obsluhy

3. So zvyšovaním úrovne spoľahlivosti narastajú aj náklady na vývoj takéhoto spoľahlivého systému. Aké sú dve hlavné príčiny za týmito nákladmi?

Drahší vývojári, techniky vývoja

Extra validácia

4. *Vymenujte tri základné modely softvérového procesu.

Waterfall (vodopádový)

Incremental (iterative) development

Integration and configuration (z existujúcich komponentov)

5. Pri inkrementálnom vývoji je menšie riziko zlyhania celého projektu. Aké sú dva hlavné dôvody?

Používanie – najžiadanejšia funkcionálna ide prvá a už ju zákazník používa od začiatku

Testovanie – najžiadanejšia/najdôležitejšia funkcionálna sa teda aj najviac otestuje

6. Aké sú tri hlavné aktivity projektového riadenia?

Plánovanie projektu

Spĺňanie požiadaviek/specifikácie

Riadenie rizík

Odhady nákladov

Zabezpečenie techniky (HW)

Riadenie ľudí

7. Ktoré tri základné veci musí obsahovať projektový plán?

Míľniky

Časový plán (kedy)

Analýza rizík

Rozdelenie úloh(čo)

Náklady

Zdroje (kto)

8. Aké sú dve hlavné plánovacie aktivity pri agilnom vývoji z pohľadu projektového plánovania?

Iteration planning (plánovanie najbližšieho kola)

Release planning(plánovanie dodávky)

9. Aké sú hlavné riziká v projektovom riadení vývoja softvéru?

Projektové

Produktové

Podnikateľské

10. Čo popisuje dokument s používateľskými požiadavkami a dokument so systémovými požiadavkami?

User reqs – špecifikácia pre zákazníka

System reqs – detailná (rozšírenie user reqs) špecifikácia pre vývoj

11. Uved'te aké sú hlavné kategórie nefunkcionálnych požiadaviek pri ich delení a uved'te príklad z každej kategórie

Product (produktové)

Organizational (organizačné – zákazník aj vývojár)

External (externé)

12. Ktorým UML diagramom sa dá zakresliť architektúra systému?

Diagram tried

Nie UML: box and lines, entito-relačný

13. Uved'te možné architektonické spôsoby ukladania a zdieľania údajov (OVER RIT!!)

Centrálné úložisko

Distribúované (každý syst. má vlastné) úložisko

14. Uved'te možné architektonické prístupy na delenie systémov (na menšie časti)

Objektový model

Data-flow / pipeline / pipe and filter model

15. Uved'te možné spôsoby centrálneho riadenia subsystémov

Call-return model (sekvenčné riadenie)

Manager model (paralelné riadenie)

16. Uved'te možné spôsoby riadenia sub-systémov pri riadení prostredníctvom udalostí

Broadcast model (rozposielanie udalostí)

Interrupt-driven model (prerušená)

17. Vymenujte aspoň dva z troch generických modelov softvérového procesu

Vodopádový model (The waterfall model)

Evolučný vývoj (Evolutionary development)

Vývoj založený na existujúcich komponentoch (Componentbased software engineering)

18. Ktoré pohľady (perspektívy) treba zohľadniť pri návrhu architektúry systému?

logical

process

physical

development

19. Vymenujte štyri základné aktivity softvérového procesu

špecifikácia

návrh a implementácia

validácia

evolúcia

20. Spárujte nasledovné príklady systémov podľa typu ich vážnosti

Systém na správu účtov v banke - Business-critical system

Navigačný systém vesmírnej lode - Mission-critical system

Riadiaci systém v jadrovej elektrárni - Safety-critical system

21. Ktoré fundamentálne aktivity sú spoločné pri každom softvérovom procese?

Requirements analysis and definition

System and software design

Implementation and unit testing

Integration and system testing

Operation and maintenance

Software specification

Software design and implementation

Software validation

Software evolution

22. Označte nasledovné požiadavky podľa ich typu:

F Systém musí archivovať originály fotografií, ktoré používateľ zmení

F Systém musí umožniť používateľovi vyhľadávať fotky podľa miesta odfotenia

N Systém musí spĺňať požiadavky paragrafu 7 zákona 428 / 2002 Z.z.

F/N Systém musí poskytnúť komfortný prehliadač fotiek

N Systém musí byť spustiteľný na počítačoch aj mobilných zariadeniach ← je to správne???

Ano, pozri <https://goo.gl/41fXq2> // mas viacero takychto opravenych testov??, mozes ich pridať hore na zaciatok docu
nemam

23. Ktoré diagramy patria do ktorého systémového modelu? Poprepájajte.

Kontextový model: Diagram architektúry

Model údajov: Entitno-relačný diagram

Model správania: Stavový diagram

Diagram použitia (use-case)

Diagram toku údajov (data-flow)

Sekvenčný diagram

Diagram aktivít

24. Čo je to model? Čo je model softvérového procesu?

Model= súbor objektov reprezentovaných PC systémom spolu s ich vlastnosťami a vzťahmi

Model softvérového procesu= Zjednodušená reprezentácia procesu softvéru, vychádzajúca (vyplývajúca) z určitého špecifického pohľadu.

25. Vymenujte základné atribúty, ktoré by mal mať dobrý softvér.

Maintainability(udržiavateľnosť), Dependability (spoľahlivosť), Efficiency (účinnosť), Acceptability (akceptovateľnosť)

26. Ktoré problémy existujú pri použití prirodzeného jazyka v špecifikácii požiadaviek?

Nedostatok jasnosti, Zmätky v požiadavkách, Zlúčenie požiadaviek

27. Aké diagramy (UML alebo iné, uveďte aspoň po jednom príklade) sa používajú na popis sémantického dátového modelu?

Abstract Hierarchy diagram

28. Čo je to proces? Čo je to softvérový proces?

Proces= inštancia programu vrátane všetkých hodnôt premenných a stavu

Softver.proces= Súbor činností, ktorých cieľom je rozvoj alebo vývoj softvéru

29. Vymenujte 4 hlavné vlastnosti (atribúty) softvérového systému spadajúce pod spoľahlivosť.

Spoľahlivosť, bezpečnosť, ochrana, dostupnosť, chybová tolerancia, opraviteľnosť, udržiavateľnosť, pružnosť

30. Vymenujte alternatívy k prirodzenému jazyku, ktoré slúžia na špecifikáciu systémových

Ways of writing a system requirements specification



Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

požiadaviek.

30/10/2014

Chapter 4 Requirements Engineering

43

31. Aké diagramy sa používajú na popis architektúry SW? Uved'te aspoň 1 UML a aspoň 1 nie-UML diagram.

UML: Class Diagram

nie UML :Box a line diagramy

Doplniť:

1. Vymenujte, ktoré základné vlastnosti musí spĺňať spoľahlivý (dependable) system (4)

Maintainability

Dependability and security

Efficiency

Acceptability

//nemalo by to byť?// to je to iste skoro

- Dostupnosť
- Spoľahlivosť
- Bezpečnosť
- Zabezpečenie
- Životaschopnosť

2. Aký je rozdiel medzi dvomi spoľahlivosťami (dependability vs reliability)?

reliability - pravdepodobnosť, že systém korektne doručí zákazníkovi požadovaný softvér

dependability - skladá sa z ďalších vecí, patrí tam napr. (reliability, safety, security,...)

Skúška z pred 2 rokov

1. Aký je rozdiel medzi dvomi oblasťami spoľahlivých systémov?

- **Security:**

- schopnosť ochrániť systém pred náhodným alebo zámerným vonkajším útokom

- **Safety:**

- *Bezpečnosť je vlastnosť systému, ktorá odráža schopnosť systému pracovať, normálne alebo abnormálne, bez nebezpečenstva spôsobenia zranenia alebo usmrtenia človeka a aby nedošlo k poškodeniu životného prostredia.*

2. Aké sú dve hlavné výhody implementácie najžiadanejšej/najprioritnejšej funkcionality ako prvej pri inkrementálnom vývoji?

- *hlavná funkcionality softvéru je implementovaná ako prvá a softvér vieme aj reálne využívať*
- *hlavná funkcionality sa tým pádom aj najdlhšie testuje*

3. Ktoré tri základné veci musí obsahovať projektový plán?

- *časový plán (kedy)*
- *zdroje (kto)*
- *rozdelenie úloh (čo)*

4. Aké tri triedy projektových rizík poznáte?

- *projektové*
- *produktové*
- *podnikateľské (obchodné, biznis)*

5. Prepíšte cieľ na materiálnu požiadavku: Vyhľadávanie v údajoch má byť rýchle

- *transakcie/s, reakčný čas*
- *napr. odozva na vyhľadávanie do 3s pri 1000 súčasných požiadavkách*

6. Aký je rozdiel medzi používateľskými a systémovými požiadavkami?

- **používateľské:**
 - *sú písané v prirodzenom jazyku*
 - *diagramy služieb ktoré poskytuje systém a ich operačné obmedzenia*
- **systémové:**
 - *štruktúrovaný dokument popisujúci detailný opis systémových funkcií, služieb a operačných obmedzení*
 - *Definuje čo má byť implementované, takže môže byť časť kontraktu medzi klientom a dodávateľom*

7. Aké dva architektonické štýly poznáte na centralizované riadenie?

- *manager model*
- *call-return model*

8. Aké dva architektonické štýly poznáte na riadenie prostredníctvom udalostí?

- *broadcast model*
- *interrupt-driven model*

9. Uved'te aspoň 3 princípy agilných metód

- *Zamerajte sa na kód, skôr než navrhovanie;*
- *Sú založené na iteratívnom prístupe k vývoju softvéru;*
- *Sú určené na rýchlu dodávku pracovných programov a vyvíjať ich tak rýchlo, aby spĺňali meniace sa požiadavky.*

10. Čo je to refaktORIZÁCIA?(refactoring)

- *Je proces zlepšovania kódu, kde je kód reorganizovaný a prepísaný, aby bol efektívnejší*

11. Akým spôsobom sa rieši testovanie v XP?(stačí názov metódy)

- *Test-first vývoj*
- *Písanie testov pred kódom upresňuje požiadavky, ktoré majú byť realizované.*

12. Aký je rozdiel medzi konfiguráciou softvéru v čase dodania a v čase návrhu?

- **Deployment-time configuration:** všeobecné (generic) systém je nakonfigurovaný vložení znalostí požiadaviek zákazníka a obchodných procesov(?). Softvér sám v sebe sa nezmení
- **Design-time configuration:** obyčajný generic kód je spravený a zmenený podľa požiadaviek jednotlivých zákazníkov

13. Čo popisuje a načo sa dá využiť návrhový vzor?

- je to znovu použitie abstraktných vedomostí a problému(?) a jeho riešení
- vzor by mal byť dostatočne abstraktný aby mohol byť použitý v rôznych situáciach
- vzor sa často spoliehaako dedičnosť a polymorfizmus //pičovina
- predstavuje všeobecné riešenie problému , ktoré sa používa pri návrhu počítačového programu(nie je knižnica ,ani časť kódu , je to abstraktný postup)

14. Aký je rozdiel medzi validovaním softvéru a testovaním defektov?

- **validačné testovanie:** demonštrovať tvorcom a zákazníkom, že softvér splňuje požiadavky
- **chybové (defekt) testovanie:** zistenie chýb a kazov v softvéri, kde správanie programu nie je správne, alebo nie je v súhlade so špecifikáciou

15. Správne priradiť kategóriu testov (A-whitebox, B-blackbox, C-iné)

- Release testing - C
- Štruktúrne testovanie - A
- Partition testing - B
- Path testing - A

Skúška spred x rokov

1. Ako môže operátor ovplyvniť systém tak, že sa systém stane nebezpečným pre zdravie alebo unsafe, napriek tomu, že systém certifikovaný ako spoľahlivý (reliable)?

- Nesprávne používa

2. Prečo je dôležité počas fázy špecifikácie požiadaviek spísať používateľskú aj systémovú špecifikáciu?

- Aby sa predišlo tomu, že zákazník dostane produkt, ktorý nespĺňa jeho požiadavky. Požiadavky by mali byť súčasťou kontraktu pre vývoj softvéru. Preto je podstatné, aby tieto požiadavky boli tak úplné, ako sa dá. (aj používateľská aj systémová špecifikácia)

3. Ktoré tri základné veci musí obsahovať projektový plán?

- Rozdelenie úloh
- Zdroje
- Časový plán

4. Konfiguračný manažment zahŕňa:

- dokumentovanie systémových nastavení
- periodickú údržbu
- baseline systémy

5. Kategorizujte požiadavky na funkcionálne (F), nefunkcionálne, organizačné (NF-org), nefunkcionálne produktové (NF-prod), a nefunkcionálne externé (NF-ext)

- Systém musí byť naprogramovaný v Java - NF-org
- Systém musí umožniť používateľovi vyhľadávať fotky podľa miesta odfotenia - F
- Systém musí spĺňať požiadavky 7. zákona 4.28/2002 2.2 - NF-EXT
- Systém musí mať dostatočne rýchlu odozvu pri vyhľadávaní - NF-prod

- *Systém musí byť dostupný na počítačoch aj mobilných zariadeniach - NF-PROD*

6. Chovanie systému sa dá dynamicky popísať o dvoch pohľadoch, ktoré to sú?

- *Spracovanie dát*
- *Stavové modely*

7. Kedy by ste použili na spracovanie udalosti systém rozposielania (broadcast) a kedy prerušenia (interrupts)?

- **Broadcast** – *účinný pri integrácii subsystémov na rôznych počítačoch v sieti*
- **Interrupts** – *používa sa v real-time systémoch, kde rýchla odozva na udalosť je nevyhnutná*

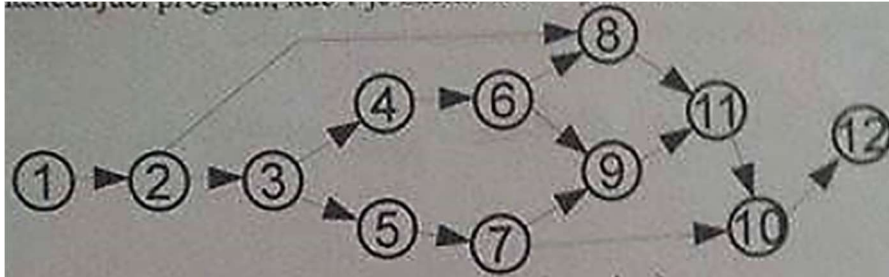
8. Akými metódami sa bežne identifikujú objekty zo špecifikácie pri objektovo-orientovanom návrhu? Objekty sa identifikujú podstatnými menami (neviem ako sa volá tá metóda oficiálne)

9. Ako je vyriešené testovanie v XP, kde neexistuje špecifikácia a teda ani testovacie prípady z nej vyplývajúce? Napíšte názov takéhoto testovania aj ho opíšte.

??

asi zapojenie užívateľov v oblasti vývoja a validácie testu

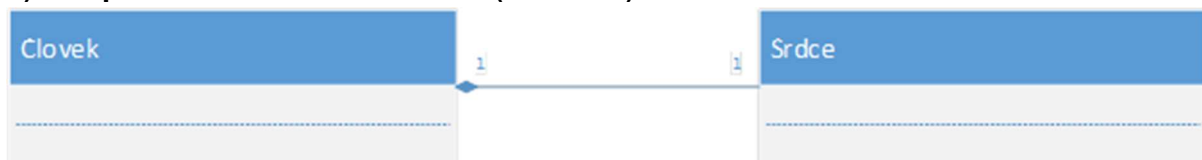
10. Otestujte nasledujúci program, kde 1 je začiatok a 12 je koniec:



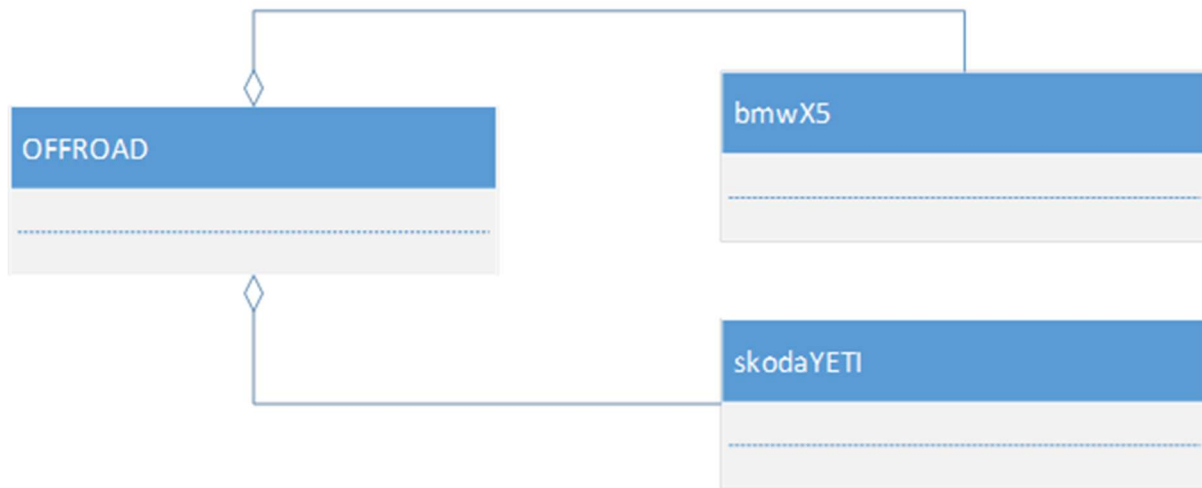
- a) Identifikujte cesty pre metódu testovania ciest (path testing)
 - 1,2,3,5,7,10,12
 - 1,2,3,4,6,9,11,10,12
 - 1,2,8,11,10,12
 - 1,2,3,5,7,9,11,10,12
 - 1,2,3,4,6,8,11,10,12
- b) Označte (podčiarknite) tie cesty identifikované v časti (a), ktoré by sa testovali v prípade použitia metódy štruktúrovaného testovania (structure testing)
-nemali by sa označiť cesty tak, aby sme otestovali každú "nodu" aspon raz

12. použite gramaticky správnu UML notáciu a zobrazte:

- a) kompozíciu tried srdce a človek ("vlastní")



- b) agregáciu tried bmwX5 a skodaYETI do triedy OFFROAD ("patrí")



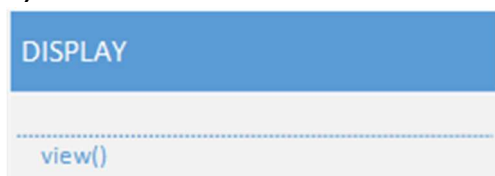
- c) prípad použitia VYTVOR



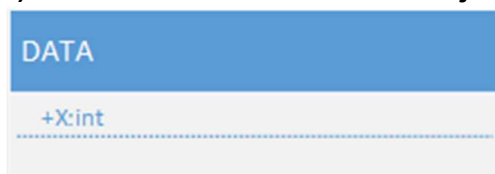
- d) objekt FEI v triede STU



- e) triedu s názvom DISPLAY obsahujúcu metódu view()



- f) triedu s názvom DATA obsahujúcu verejný atribút X typu integer



13. Nakreslite UML diagram tried a zobrazte:

- človek má meno, email, telefónne číslo
- Učiteľ a študent sú ľudia (špecifické typy človeka)
- adresa má ulicu, číslo domu, mesto, PSČ
- človek má práve jednu adresu, ale na jednej adrese môže žiť niekoľko ľudí
- učiteľ učí maximálne 2 predmety
- predmet môže navštevovať ľubovoľný počet žiakov

Použite UML notáciu. Orámujte a pomenujte diagram. Zobrazte všetky násobnosti (multiplicity)
Pomenujte všetky asociácie v diagrame