PB007 – ANalýza A NÁvrh Systémů

(vypracované teoretické otázky CZ)

Český překlad vypracovaných teoretických otázek od T.F. verze 3.2.2011

(omlouvám se za případné chyby)

Otázky, které jsou uzávorkované byly dopsány T.F., buď proto, neboť je pan doktor Jaroslav Ráček vzpomenul na přednášce, že se na ně zvykne ptát a nebo jednoduše proto, že se T.F. zdáli důležité...

Tímto bych chtěl taky T.F. poděkovat za vypracovaný materiál a doufám, že bude dál dobře sloužit, ostatním good luck u zkoušky ;-) M.H.

[Co je to software ?]

Software je celkový souhrn počítačových programů, procedur, pravidel, doprovodné dokumentace a dat, které patří k provozu počítačového systému.

[Jak se dělí software podle toho, komu je určený? Charakterizujte.]

Generické produkty – krabicový software, samostatné systémy prodávané na volném trhu libovolnému zákazníkovi, který si může výrobek koupit.
Smluvní/zakázkové/zákaznické produkty – systémy objednané určitým zákazníkům, dodavatel zakázky je smluvně vázaný, specifikace produktu je důležitou součástí kontraktu.

1. Popište model životního cyklu „vodopád", jeho etapy a jejich výstupy.

Analýz - počáteční etapa představuje studium problému před jeho řešením, shromažďuje požadavky na vyvíjený systém. Výstupem je specifikace systému.
Návrh - jeho cílem je identifikovat základní jednotky/moduly a jejich rozhraní. Z výsledků návrhu dostáváme na výstup podklady k programování.
Implementace - návrh systému je transformovaný do formy programů, výstupem je spustitelný kód.
Testování - jeho cílem je ověřit, že implementace systému splňuje požadavky stanovené při jeho analýze. Výstupem je odladěný spustitelný kód.
Provoz a údržba - systém je nasazený u zákazníka, může být modifikovaný na základě dodatečně zjištěných nedostatků nebo změn v okolí.

2. Jaké modely se dají použít při obrysové specifikaci? Popiš.

Model „*inkrementálního*" životního cyklu – projekt je rozdekomponovaný na malé části – dobře zvládnutelné, postupně se odevzdávají zákazníkovi a skládají do výsledného systému, na druhé straně z toho plyne problém s čistotou návrhu a konstrukce.
Model životního cyklu „*prototypování*" – v tomto modelu jsou nástrojem na získání požadavků prototypy, na kterých se zkouší budoucí funkcionalita. Po specifikaci je prototyp zahozený.
Model životního cyklu „*výzkumník*" – metoda „pokus-omyl", tj. experimentování, u kterého se často netuší, jak dopadne. Je náročný na manažerské řízení a není možné nahradit řešitele.

3. Napište Lehmanovy „zákony". (1980,1985)

Zákon trvalé proměny - systém používaný v reálném prostředí se neustále mění, dokud není levnější systém restrukturalizovat, nebo nahradit zcela novou verzí.
Zákon rostoucí složitosti - Při evolučních změnách je program stále méně strukturovaný a vzrůstá jeho vnitřní složitost. Odstranění narůstající složitosti vyžaduje dodatečné úsilí.
Zákon vývoje programu - Rychlost změn globálních atributů systému se může jevit v omezeném časovém intervalu jako náhodná. V dlouhodobém pohledu se však jedná o seberegulující proces, který lze statisticky sledovat a předvídat.
Zákon invariantní spotřeby práce - Celkový pokrok při vývoji projektů je statisticky invariantní. Jinak řečeno, rychlost vývoje programu je přibližně konstantní a nekoreluje s vynaloženými prostředky.
Zákon omezené velikosti přírůstku - Systém určuje přípustnou velikost přírůstku v nových verzích. Pokud je limita překročena, objeví se závažné problémy týkající se kvality a použitelnosti systému.

[Co říká Brooksov zákon ?]

Přidání řešitelské kapacity u opožděného projektu může zvětšit jeho zpoždění.

[Zhodnoťte následující část specifikace, co je ní špatně ?]

Mřížka editoru – má pomáhat při umisťování entit na obrázku. Uživatel může mřížku zapnout v centimetrech a nebo palcích pomocí volby na ovládacím panelu. Na začátku je mřížka vypnuta. Mřížka může být zapnuta nebo vypnutá kdykoliv během editování a může být přepnutá kdykoliv mezi centimetry a palci. Mřížka bude k dispozici i při „zmenšení na velikost okna", ale počet čar mřížky bude redukovaný, aby nedošlo k překrytí zmenšeného obrazu hustou mřížkou.
Kritika – směs konceptů a implementačních aspektů, neúplný popis předvolaného stavu, špatně členěné, raději do kapitol, výklady doplnit o smysl…(viz- sada2, slide 37)

[Popište komponenty DFD a konvence jmen.]

Procesy – Proces ukazuje část systému, která transformuje určité vstupy na výstupy. Proces je pojmenován jediným slovem, frází nebo jednoduchou větou. Jméno vyjadřuje, co proces dělá.
Toky – Tok znázorňuje cestu, po které se pohybují datové shluky (informační pakety) z jedné části systému do druhé. V některých případech vyjadřuje pohyb fyzických materiálů. U reálných systémů mohou být na DFD současně toky, které vyjadřují pohyb materiálu a dat.
Paměti – Paměť modeluje kolekci dat v klidu. Jméno je voleno obvykle jako množné číslo jména, kterým jsou označeny pakety na tocích vedoucích do a z paměti.
Terminátory – Terminátor reprezentuje externí entity, se kterými systém komunikuje, jméno je název entity.

4. Co je esenciální a implementační paměť ? Jaké jsou vlastnosti paměti na DFD ?

Esenciální paměť - data předávaná mezi dvěma a více procesy pracujícími v různém čase.
Implementační paměť – není nutná z podstaty věci(ale nepřekáží), teoreticky by to fungovalo i bez ní, může se použít kvůli omezení hardware, jako záloha při nespolehlivých technologiích, jako rozhraní, když dva různé týmy dělají dva procesy, které spolu komunikují nebo když analytik předvídá nějakou budoucí funkcionalitu.
Vlastnosti paměti - Paměť je pasivní částí systému, data nejsou přenášena do/z paměti, pokud o to proces explicitně nepožádá. Čtení je nedestruktivní, tj. paměť se nemění, když paket s informací putuje po výstupním toku z paměti.

[Může být paměť esenciální a implementační zároveň ?]

Mezi dvěma procesy určitě ne. Ve všeobecnosti ano, z pohledu dvou může být implementační, ale když přidáme další, pro ten může být esenciální.

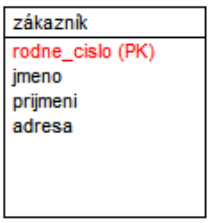
[Popište Hierarchii DFD.]

Kontextový diagram – je jen jeden, reprezentuje pohled na celý systém, obsahuje jen jeden proces.
DFD 0. Úrovně (systémové DFD) – rozdekomponovaný kontextový diagram, reprezentuje pohled na hlavní funkce a na rozhraní mezi těmito funkcemi.
DFD X. úrovně – rozdekomponování vybraných procesů na x. úrovni, na každé úrovni by mělo být 5±2 procesy+paměti.
Minispecifikace – každý proces nejnižší úrovně má právě jednu minispecifikaci, popisuje logiku procesu a pravidla transformace datových toků.

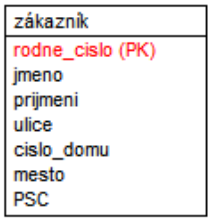
5. Popište 1. NF, uveďte vlastní příklad, který porušuje 1. NF, zdůvodněte proč a převeďte jej do správné formy.

Def. 1: Datový záznam je v 1. NF, když se v záznamech nevyskytují opakující se skupiny položek.
Def. 2: Datový záznam je v 1. NF, když jsou všechny jeho komponenty atomické(tj. atributy nejsou složené datové struktury).

Špatně:



Správně:



Adresa zákazníka je složená datová struktura a musí být tedy dekomponována na více atomických atributů.

[Popište, co je funkční závislost ?]

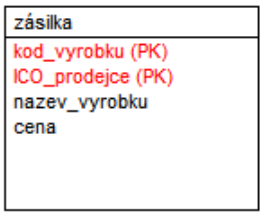
Datová položka B záznamu R funkčně závisí na datové položce A z R, pokud v libovolném časovém okamžiku každé hodnotě A odpovídá nejvýše jedna sdružená hodnota B v záznamu R. (A identifikuje B)

6. Popište 2. NF, uveďte vlastní příklad, který porušuje 2.NF, zdůvodněte proč a přetvořte jej do správné formy. Popište plně funkční závislost.

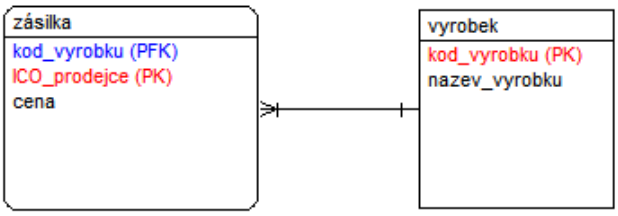
Def.: Datový záznam je v 2. NF, pokud je v 1. NF a každá neklíčová položka (atribut) v záznamu je plně funkčně závislá na každém kandidátním klíči.

2. NF vyžaduje, aby hodnoty atributů jednotlivých entit závisely na hodnotách jejich klíčů, když jde o složený klíč, tak zároveň musí být hodnoty ostatních atributů závislé na celém klíči.

Špatně:



Správně:



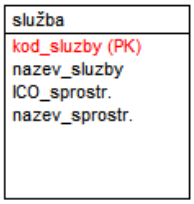
Cena výrobku závisí jak na kódu výrobku, tak i na IČu prodejce, tedy se neporušuje 2. NF; Ale název výrobku závisí jenom na jeho kódu bez ohledu na to, kdo jej dodává a tedy jde o funkční závislost jen na části klíče, z toho plyne, že je porušena 2. NF.

Plná funkční závislost - Datová položka nebo kolekce datových položek B záznamu R jsou plně funkčně závislé na kolekci datových položek A záznamu R , pokud je B funkčně závislé na celém A, ale nikoliv na vlastní podmnožině A.

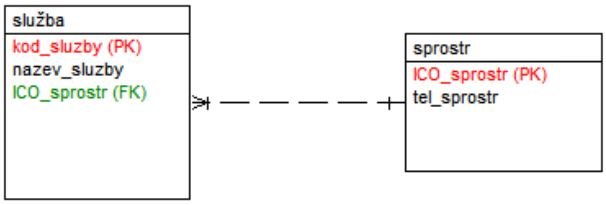
7. Popište 3. NF, uveďte vlastní příklad, který porušuje 3. NF, zdůvodněte proč a přetvořte jej do správné formy.

Def. 1: Datový záznam R je ve 3.NF, pokud je ve 2.NF a každá neklíčová položka (atribut) R je netranzitivně závislá na každém kandidátním klíči z R.
Def. 2: Záznam je ve 3.NF, pokud je ve 2.NF a každý atribut je funkčně závislý na klíči a pouze na klíči.

Špatně:



Správně:

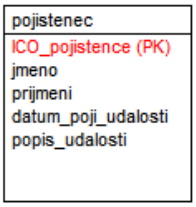


V entitní množině na obrázku existuje tranzitivní závislost, protože název zprostředkovatele závisí na IČO zprostředkovatele, které dále závisí na kódu služby. Název zprostředkovatele tak nezávisí jen na klíči, ale i dalším neklíčovém atributu.

8. Popište 4. NF, uveďte vlastní příklad, který porušuje 4. NF, zdůvodněte proč a přetvořte jej do správné formy.

- 4.NF odstraňuje v datových záznamech podmíněné funkční závislosti. Smyslem 4. NF je zajistit, aby se v databázi nevyskytovali entity, které nemají přiřazené hodnoty některých atributů.

Špatně:



Správně:

V uvedené entitní množině dochází k porušení 4. NF v případě atributů, které uchovávají data o pojistné události, které se vyplňují až v okamžiku, když se stane, což ovšem nemusí nastat. Podmíněné atributy jsou tedy přesunuté do samostatné entitní množiny.

9. Popište vyvažování DFD a DD.

- Každý datový tok a každá datová paměť na DFD musí být definovány v datovém slovníku. Pokud v něm chybí, jsou považovány za nedefinované.
- Každý datový element nebo paměť definované v datovém slovníku se musí vyskytovat někde na DFD. V opačném případě jsou považovány za přebytečné, za nepoužité v systému. Obvykle jsou přebytečné pojmy v DD důsledkem úprav prvotních modelů.

10. Popište vyvažování DFD a minispecifikace.

- Každý proces na DFD musí být asociován s DFD na nižší úrovni nebo se specifikací procesu, ale nikoliv současně s oběma. Pokud existují obě definice procesu, je model zbytečně a nebezpečně redundantní.

- Každá specifikace procesu musí mít sdružený proces na nejnižší úrovni. Pokud chybí procesy, pak byly obvykle zrušeny při úpravách DFD.

- Musí souhlasit vstupy a výstupy. Vstupní a výstupní toky stejně jako paměti zakreslené u procesu na DFD by měly mít odpovídající operace ve specifikaci (GET, INPUT, DISPLAY ...).

[Popište vyvažování DD a minispecifikace.]

Pro každý odkaz ve specifikaci procesu (obvykle podstatné jméno) musí platit jedna z následujících skutečností:

- Souhlasí se jménem datového toku nebo datové paměti připojené k procesu, který minispecifikace popisuje.

- Může to být lokální název, explicitně definovaný ve specifikaci.

- Položka v datovém slovníku je komponentou datového toku nebo datové paměti připojené k procesu.

11. Popište vyvažování CDFD a STD.

- Pro každý řídící proces existuje stavový diagram jako jeho specifikace. Ke každému stavovému diagramu musí existovat řídící proces na DFD.

- Každá podmínka ve stavovém diagramu musí odpovídat nějakému vstupnímu řídícímu toku, který vede do řídícího procesu sdruženého s příslušným STD.

- Každé akci stavového diagramu musí odpovídat nějaký výstupní řídící tok řídícího procesu sdruženého se tímto STD.

12. Popište vyvažování ERD, DFD a minispecifikace.

- Paměť na DFD musí odpovídat objektovému typu, relaci nebo jejich kombinaci (asociovanému objektu) na ERD. Výskyt paměti bez odpovídajícího objektového typu nebo objektový typ bez odpovídající paměti jsou považovány za chybu.

- Jména objektových typů na ERD a datových pamětí na DFD musí odpovídat. Konvence předpokládá použití množného čísla na DFD a jednotného čísla na ERD.

- Položky v datovém slovníku musí být aplikovatelné současně na DFD i na ERD. Datový slovník musí obsahovat současně definici objektu z ERD i paměti z DFD.

Př.: ZÁKAZNÍCI = {ZÁKAZNÍK} množina instancí={instance}
ZÁKAZNÍK = jméno + adresa + tel.číslo + . . .

- V DFD musí existovat procesy, které přiřadí hodnoty každému datovému elementu, který je atributem objektového typu. Toto platí pro všechny instance. Existují procesy, které hodnoty čtou.

[Jaké přístupy se používají při dekompozici systému ?]

Funkčně orientovaný přístup - Systém jako množina interagujících funkcí. Funkční transformace jsou umístěny v procesech, které jsou propojeny datovými a řídícími toky.
Datově orientovaný přístup - Hledá fundamentální datové struktury aplikace. Funkční stránka (tj. různé transformace dat) je méně podstatná. Datový model definuje konceptuální model pro DB systému.
Objektově orientovaný přístup - Systém jako množina interagujících objektů, operace působící na objekty jsou zapouzdřeny v samotných objektech.

13. Popište postup analýzy metodou SASS(DeMarco). Charakterizujte tuto metodu. Uveďte výhody/nevýhody. Jaké systémy(nástroje) SASS používá ?

SASS – používá dekompozici „shora-dolů", silně funkčně orientovanou. Analýza pomocí 4 modelů:

1. Studie stávajícího fyzického systému
2. Odvození logického ekvivalentu stávajícího systému
3. Odvození nového logického systému
4. Odvození nového fyzického systému

Nevýhodou dekompozice „shora-dolů" je ve všeobecnosti, že rozčlenit systém na rozumné celky nemusí být lehké. Tato metoda to dělá postupným zjemňováním, což je sice zdlouhavé, ale výsledek je velmi dobrý.
Největší slabinou je, že na modelování dat používá jen DD a ne ERD.
Nástroje SASS – DFD, DD, strukturovaná angličtina, rozhodovací tabulky a rozhodovací stromy.

14. Popište postup logického modelování.

(Gane, Sarson (1979))

1. Vytvoření prvotního systémového DFD
2. Načrtnutí datového modelu
3. Analýza entit a jejich vztahů - logický datový model
4. Relační datový model, normalizace

5. Překreslení DFD podle datového modelu – logický procesní model.
6. Dekompozice logického procesního modelu na procedurální jednotky.
7. Specifikace detailů každé procesní jednotky.

15. Porovnejte SASS a Logické modelování, co mají společné (3 vlastnosti) a v čem se liší (3 vlastnosti).

Obě metody používají dekompozici „shora-dolů" (1), i postupy procesně orientované přístupu (DFD) (2), zahrnují i oblast návrhu (3).
Rozdílné je, že v logickém modelování se více používá datový pohled, modeluje jej pomocí ERD a v celku hlavní úlohu hraje tvorba datového modelu (1). SASS kvantifikuje ceny a termíny (2). Logické modelování se zaobírá DFD 2x, SASS 4x (3).

16. Definujte VPA a její použití.

Pohledová analýza (VPA) – jako první začala používat metodu „zdola-nahoru", používala atypické modelovací nástroje (pozorovací body, pohledy, tabulkový diagram akcí, akční diagram…).

Používá se když:
- hierarchie entit není dosud vytvořena,
- hierarchie entit není na první pohled zřejmá,
- systém není přirozeně hierarchicky uspořádán.

17. Porovnejte DeMarcovu (SASS) a Yourdonovu metodu (YMSA), 5 společných vlastností.

Obě patří mez metody strukturované analýzy (1), používají DD (2) i DFD (3), jsou procesně orientované (4), u obou začíná proces analýzou současného systému (5).

18. Jaké druhy událostí jsou v YMSA ? Popište je. Uveďte 3 vlastní příklady.

F (flow) - tokově orientovaná událost, je sdružená s datovým tokem (příjem jednoho nebo několika datových paketů)
T (tempoval) - časová událost, nastává v nějakém významném časovém bodě (absolutní nebo relativní čas)
C (control) - řídící událost, povel, signál (je sdružena s vnějším řídícím tokem, povelem)

Příklad:
U1: Zákazník vystavuje objednávku. (F)
U2: Zákazník ruší objednávku. (F)
U3: Vedení požaduje zprávu o prodeji každé ráno. (T)
U4: Pokyn k vyhodnocení stavu a vývoje prodeje. (C)

[Jaké varianty jsou možné při tvorbě modelů okolí ?]

- Na základě informací od uživatelů lze sestavit kontextový diagram. Zkoumáním terminátorů a toků odhadneme události. Prověříme pomocí dalších modelů (procesní, datový m.).
  Kontextový diagram => Seznam událostí => ...

- Začneme s datovým modelem - ERD. Nalezneme esenciální objekty a jejich vztahy. Pro každou entitu hledáme, jaké vnější události vedou k jejímu použití, změně atd. Sestavíme předběžný seznam událostí. Pomocí něho vytvoříme kontextový diagram.
  ERD => Seznam událostí => Kontextový diagram => ...

19. Jak se v YMSA dělá prvotní DFD z modelů okolí, co nejpodrobněji rozepište postup. Co obsahuje model okolí ?

- Pro každou odezvu na událost ze seznamu událostí zakreslíme do prvotního DFD jeden proces.
- Proces pojmenujeme podle očekávané odezvy na tuto událost.
- Zakreslíme datové paměti, které modelují data nezbytná pro zpracování asynchronně probíhajících událostí.
- Doplníme odpovídající vstupní a výstupní toky.
- Vytvořený DFD ověříme proti kontextovému diagramu.

Výchozí podklady - *Model okolí systému*, který obsahuje dokumety:

- kontextový diagram,
- seznam událostí,
- dokument o účelu systému,
- zahájena tvorba datového slovníku (datová rozhraní mezi systémem a terminátory)

20. Co je to koheze a propojení v strukturovaném návrhu ?

Koheze – je míra, která vyjadřuje provázanost úloh řešených pomocí daného modulu.
Propojení - je míra vzájemné propojenosti nebo provázanosti modulů v programu.

21. Jaký je postup při modelování podle JSP(Jackson Structured Programming), uveďte příklad notace a modelu.

JSP – je metoda modulárního návrhu, návrh na základě datových struktur,
Notace je shodná pro strukturu dat i pro strukturu programu. Hlavní řídící metoda je nahoře, dále se čte shora-dolů a zleva-doprava.

Obrázky: prezentace an_09_strukt_navrh.pdf, strana 15-17

22. Co je transformační analýza, kdy se používá, uveďte postup. Co je výstupem a co vstupem ?

- Je technika, jejíž cílem je najít na DFD)vstup) skupiny procesů, které představují transformační centrum a přinést jej v podobě modulů do diagramu struktury systému (výstup).
- Používá se, když se procesy dat nějakým způsobem mění/transformují.
- Postupuje se tak, že na DFD se jde nejprve po vstupních tocích směrem dovnitř diagramu tak dlouho, dokud data nejsou použité k samotnému zpracování. První hrana před zpracováním se označí značkou. Podobně se postupuje i při výstupních tocích. Tentokrát však proti směru toku dat, postupně dovnitř tak dlouho, dokud jsou výstupní data jen formátované. V místě zpracování se první hrana po zpracování označí značkou. Všechny značky se následně spojí a ohraničený podgraf tvoří centrální transformaci DFD.

23. Co je to transakční analýza, kdy se používá ? Uveďte postup.

- Je technika, jejíž cílem je vymezit transakční centrum na DFD.
- Používá se, když je třeba určit proces, který slouží jako začínající bod akčních cest. Do struktury systému se promítne jako transakční centrum. Dílčí podgrafy na DFD se potom v diagramu struktury systému zobrazí jako následníci transakčního centra. Struktura podgrafů je následně přenesena do nižších úrovní diagramu struktury systému, buď pomocí transformační a nebo transakční analýzy, podle toho k jakému zpracování dat v podgrafu dochází.

24. Návrhové heuristiky, co to je, jak a kdy se to používá ? Vypište je a dvě podrobně rozepište.

Jsou to techniky, které slouží k optimalizaci struktury systému po její navrhnutí. Mezi nejvýznamnější patří :

H1: Vyhodnocení prvotní programové struktury s cílem redukovat propojení a zvýšit kohezi (exploze a imploze modulů).
H2: Minimalizace rozšiřujících se struktur; snaha o sbližování větví se vrůstající hloubkou.
H3: Snaha udržet rozsah efektu modulu v rozsahu vymezeném řízením daného modulu.
H4: Vyhodnocení modulových rozhraní s cílem redukovat jejich složitost a nadbytečnost a zvýšit konzistentnost.
H5: Definovat moduly, jejichž funkce je zřejmá, ale vyhnout se těm, které jsou příliš restriktivní.
H6: Hledání modulů "jeden vstup-jeden výstup", nepoužívat patologické vazby.
H7: Software balit s ohledem na daná návrhová omezení a požadavky na přenositelnost.

25. Stručně popište SSADM, jaké 3 základní druhy diagramu používá ?

SSADM – zahrnuje analýzu i návrh, metoda „zdola-nahoru", člení projekt na malé, dobře definované aktivity a specifikuje sekvence a interakce těchto aktivit.

Základní diagramy:

ERD - Logické datové struktury - ukazují, která informace je ukládána a jaké jsou vzájemné vztahy mezi jednotlivými informacemi
DFD - Diagramy datových toků - ukazují, jak se předává informace v systému
ELH - Životní cykly entit - ukazují, jak se informace mění během svého života

26. Coad & Yourdon 5-vrstvý model OOA, co a k čemu to je ?

Je to objektová metoda, která modeluje svět pomocí 5-vrstvého modelu:

Obrázek: prezentace an_11_oaa.pdf strana 15

Hlavní aktivity jsou(Nejedná se o sekvenční kroky. Rozpracované aktivity lze podle potřeby střídat v různém pořadí.):

- Nalezení Tříd-&-Objektů
- Identifikace struktur
- Identifikace subjektů
- Definice atributů
- Definice služeb

27. Popište diagram komunikace a diagram posloupností (sekvenční diagram) v UML, jejich součásti, notaci tříd, nakreslete.

Jsou to interakční diagramy, mají stejnou vypovídací schopnost, proto je třeba používat vždy jen jeden z nich. Ukazují jednotlivé třídy a objekty systému, i aktér je třídou. Diagram posloupností ukazuje interakci mezi objekty uspořádanými do časové posloupnosti. Diagram komunikace ukazuje interakci objektů a jejich propojení mezi sebou.

Obrázek: prezentace an_12_UML.pdf strana 15-16

28. Uveďte dva UML modely, které zachytávají vývoj systému v čase, popište je, nakreslete a vysvětlete jak je čas zachycený.

Diagram posloupností a diagram komunikace (obr. předchozí otázka). Na diagramu posloupností je čas zachycený na svislých osách, které představují časovou osu objektů. Na diagramu komunikace je čas skrytý v očíslovaných tocích mezi objekty. (Pozn.: v strukturované analýze zachytává čas stavový diagram a diagram ELH v SSADM)

29. Popište SCRUM, co to je ?

Je to agilní metoda (tj. iterativní a inkrementální vývoj), principem je iterativní vývoj definující 3-8 fází (tzv. sprintů), z kterých každá trvá přibližně měsíc. Nedefinuje žádné konkrétní procesy, jen zavádí pravidelné každodenní schůzky vývojového týmu, kde se zhodnotí, co bylo dokončené od minule a které nové úlohy jsou teď na řadě. Každý sprint je zakončený demo ukázkou zákazníkovi, který poskytne zpětnou vazbu.

30. Metoda Select Perpetive – co to je, jaké modely používá ?

Je to projektový postup, který kombinuje procesní modelování a objektově orientované metody. Postupuje se inkrementálně. Architektura systému je založená na komponentech. Využívá UML diagramy a procesní mapy.

31. Popište WebML jaké modely používá ?

Je to modelovací nástroj i metoda, umožňuje vytvořit komplexní model webové aplikace (důležitým prvkem je navigace).
Základní modely:

- Datový model – návrh datové struktury
- Hypertextový model – tvoří jej model kompozice webové aplikace a model navigace
- Prezentační model – je chápaný jak transformace předešlých modelů do konkrétní webové prezentace