PROGRAMŲ SISTEMŲ INŽINERIJA II

Prof. A. Čaplinsko 2010m. dėstomo kurso,
„Programų sistemų inžinerija II“ paskaitų konspektas

Parengė Kęstutis Matuliauskas

2010 m. birželio 19 d.

Užrašams

Pratarmė

A. Čaplinsko skaitomam kursui – „*Programų sistemų inžinerija II*“, ligi šiol nebuvo kokio nors konkretaus, struktūrizuoto, konspekto, kurį būtų lengva suprasti, nebūtų apkrautas per dideliu kiekiu informacijos, o informacija jame būtų dėsningai išskirstyta į temas, kurių kiekviena klasifikuota pagal tam tikrus, su ta tema susijusius, dėsningumus. Šis konspektas, bent jau iš dalies, turėtų užpildyti šią spragą.

Pagrindiniai konspekto šaltiniai – A. Čaplinsko paskaitose paskelbta informacija, bei jo parengtas, šiuo metu 11 tomų, skaidrių rinkinys, sudarytas iš daugiau kaip poros tūkstančių skaidrių. Taip pat dalis informacijos yra paimta iš internetinių enciklopedijų bei kitų studentų darytų „Programų sistemų inžinerijos“ kurso santraukų.

Turinys

Terminų žodynas

Diagramų rūšys *(diagrams)*:	
• užduočių	use case
• komponentų	component
• sekų	sequence
• ansamblių	collaboration
• išdėstymo(sistemos konfigūracijos)	deployment
• statinės struktūros	class
• būsenų	state
• veiklos	activity
Reikalavimai:	
• reakcijos laikas	response time
• gaišties laikas	latency
• pralaidumas	throughput
• produktyvumas	efficiency
• masto keitimas	scalability
• paprastumas	easy-to-use
• patogumas vartotojui	user-friendliness
• informavimo priemonių tinkamumas	helpfulness
Reikalavimų statusai:	
• esminis *(privalomas)*	esiantial **(E)**
• pageidaujamas	desirable (**D**)
• papildomas	optional (**O**)
Reikalavimų galiojimo laikas:	
• pastovus	stable **(S)**
• sąlyginai pastovus	unstable (**U**)
• laikinas	temporal (**T**)
UML:	
• žymė	tag
• žymėtoji reikšmė	tagged value
• konstrukcijos rūšis	stereotype
Kiti:	
• nukirstasis perėjimas	stubbed transition
• atsakomybės juosta	swimlane
• prieigos maršrutas	pathname
• Perėjimo veiksmas	transition action
• Perėjimo sąlyga	transition guard

Trumpiniai

UML – **s**tandartinė grafinė modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus. *(„Unified Modeling Language“)*

UML – tai esminis patikimų ir lengvai prižiūrimų PS kūrimo įrankis.

OCL – tai UML objektinių ribojimų aprašymo kalba, sukurta IBM. (*„Object Constraints Language“)*

ISO – pirmaujanti pasaulyje tarptautinių standartų kūrėja. *(„International Organization for Standartization“)*

COTS – tai sistema sudaryta iš komponentų, paprastai įsigyjamų rinkoje *(„Commercial-off-the-shelf“)*

API – PĮ kūrimo sąsaja tarp žmogaus ir kompiuterio*(„application programming interface“)*

GUI – grafinė vartotojo sąsaja (*„Graphic User interface“*)

PĮ – programinė įranga

TĮ – techninė įranga

PS – programų sistema

PSGC - programų sistemos gyvavimo ciklas

PSGCM – PSGC modelis

UFK – užduočių formulavimo kalba

UPP – Užduoties pateikties protokolas

ISO 9126 – tarptautinis PĮ kokybės vertinimo standartas sudarytas.

ISO 9126 sudėtis – sudarytas iš kokybės modelio ir 3 metrikų: išorinės, vidinės ir kokybės naudojime.

Quint – tai plačiai naudojamas PĮ kokybės standarto ISO 9126 plėtinys. (*„QUality in InformatioN Technology“)*

Quint sudėtis - skiria 6 PĮ kokybės atributų klases ir 32 atributus ir apibrėžia kokybės matavimo matus.

Terminai

Modelis – tai aiškiai nusakytą paskirtį turintis supaprastintas originalo analogas.

Abstraktusis modelis - tai teisingų tvirtinimų *(teoremų)* ir galbūt teiginių, apie originalo statines ir dinamines charakteristikas, rinkinys.

Apibendrinimo asociacija – tai neformalus būdas pavaizduoti, kad viena užduotis yra panaši į kitą, bet, funkcionalumo požiūriu, yra už ją šiek tiek bendresnė.

(ANSAMBLIŲ DIAGRAMOS)

Ansamblis – ji sudaro ansamblio dalyviai ir tik to ansamblio kontekste turintys aiškiai apibrėžtą prasmę tuos dalyvius siejantys sąryšiai.

Multiobjektai - vaizduoja asociacijos "daugelis" gale esančias objektų aibes.

(KLASIŲ DIAGRAMOS)

Klasė - tai modelio konstrukcija, kuria modeliuojama atitinkama dalykinės srities esybė.

Klasė - tai objektinio projektavimo pagrindas, bendresnė poklasio versija.
Poklasis – tai patikslintą klasės versiją.

Atributai - aprašo tą informaciją, kurią klasė turi apie savo objektus.

Operacijos – tai klasės vykdomi procesai.

Abstrakti klasė – tai tik dalinai realizuota klasė, kurios funkcionalumu galima pasinaudoti tik ją išplėtus.

Kompozicija – tai agregavimo rūšis, susiejanti agregatą ir jo dalis visam jų gyvavimo laikui.

Kvalifikatorius – tai atributų sekcija.

Kvalifikuota asociacija – tai binarinė asociacija, bent viename savo gale turinti kvalifikatorių.
Asociacijos klasė – tai konstrukcija, turinti ir klasei ir asociacijai būdingus ypatumus, kitaip tariant - tai asociacija, kuri tuo pat metu yra ir klasė.

Apibendrinimo ryšys *(„is-a" ryšys)* – sieja poklasį ir klasę.

Atvirkščias ryšys – tai klasę ir poklasį siejantis ryšys.
Specializavimo ryšys – tai atvirkščias ryšis.
Poklasio ryšis – tai specializavimo ryšis.

Interfeisas – tai objekto elgsenos aprašas, pateikiamas atskirai nuo to objekto realizacijos ar būsenos. Interfeisas yra realizacijos neturinti klasė.

Asociacija - parodo, kad dvi klases sieja tam tikras ryšys.

Priklausomybė - tai silpnesnė nei asociacija ryšio forma, naudojimo ryšys, parodantis, kad kokio nors daikto specifikacijos pokyčiai padaro poveikį kitam, jį naudojančiam, daiktui.

Utilita – tai globaliųjų kintamųjų ir procedūrų grupavimo priemonė.
Metaklasės – tai tokios klasės, kurių realizacijomis yra ne objektai, bet klasės.

(OBJEKTŲ DIAGRAMOS)

Objektas - klasės realizacija.

Sudėtiniai objektai – tai objektai, sudaryti iš tampriai tarpusavyje susietų vidinių dalių.

(BŪSENŲ DIAGRAMOS)

Būsena - objekto "nuotrauka" (jo atributų reikšmių visuma) kokiu nors to objekto gyvavimo laikotarpiui priklausančiu laiko momentu.

Vidinis perėjimas – tai toks perėjimas, kuriam suveikus liekama toje pačioje būsenoje.

Sudėtinis perėjimas - turi keletą pradinių ir/arba keletą tikslinių būsenų.

Perėjimo ryšys - vaizduoja dvi objekto būsenas siejantį sąryšį

Perėjimo veiksmas *(transition action)* – tai nepertraukiama operacija, vykstanti perėjimo metu.

Perėjimo sąlyga *(transition guard)* – ji aprašo, kokia sąlyga turi būti patenkinta, kad įvyktų perėjimas. Vienu metu gali vykti tik vienas perėjimas. Tai reiškia, kad perėjimo sąlygos negali persidengti.

Įvykis – tai koks nors reikšmingas atsitikimas, galintis iššaukti perėjimą iš vienos būsenos į kitą.

(VEIKLOS DIAGRAMA)

Veiklos diagramos – tai būsenų diagramų porūšis, skirtas parodyti veiklos būsenų (būsenų, kuriose kas nors yra padaroma) sekas.

Atsakomybės juosta - tai tam tikra paketo rūšis, skirta organizuoti atsakomybes už veiklas klasės viduje.

(IŠDĖSTYMO DIAGRAMA)

Komponentai - tai vykdomi kodo blokai, teikiantys vieną ar daugiau paslaugų, kiekviena iš kurių per paprastą interfeisą pateikia tam tikrą funkcionalumą. Interfeisai išlieka nepakeisti net ir tuomet, kuomet paslaugos teikiamas funkcionalumas pakinta.
Sistemos išdėstymo mazgas - kokią nors sistemos dalį vykdantis vienetas, paprastai, kompiuteris ar kokia nors kita techninė įranga.
Jungtis - mazgų jungtys parodo sistemos komunikavimo maršrutus (interaction paths).
Sistemos išdėstymo mazgai - mazgai, vaizduojantys komponentų egzempliorius

Mazgas – tai vykdymo meto fizinis objektas naudojamas kaip vienas iš vykdančiosios sistemos resursų. Jis turi teikti bent jau atmintį, tačiau dažniausiai pateikia ir kokį nors procesorių.

(UŽDUOČIŲ DIAGRAMA)

Vartotojo tikslas – tai savo darbą siekiančio atlikti pirminio agento tikslas.

Sumarinis tikslas – tai vartotojo tikslų rinkinys.

Subfunkcija – tai potikslis arba scenarijaus žingsnis, esantis už vartotojo tiesioginių interesų ribų.

Trigeris – tai išorinis įvykis, inicijuojantis užduoties vykdymo pradžią.
Prioritetas – tai užduoties svarba, lyginant ją su kitomis užduotimis.
"Prieš" sąlygos – tai sistemos ir galbūt agentų būsenos, kurioms esant galima pradėti vykdyti užduotį.
Sėkmingos baigties sąlygos – tai sistemos ir galbūt agentų būsenos, baigus vykdyti užduotį.
Nesėkmingos baigties sąlygos – tai sistemos ir galbūt agentų būsenos, nepavykus įvykdyti užduotį.

Scenarijus – tai seka sąveikų, vykdomų su tikslu pasiekti pirminio agento tikslą, vykstančių prie tam tikrų sąlygų ir duodančių rezultatus, priartinančius prie siekiamo tikslo.

Scenarijaus sprogimas – tai atvejis, kai atsiranda per daug užduočių modelio abstrakcijos lygmenų.

UML ribojimai – tai taisyklės, aprašančios ką galima ir ko negalima daryti.
UML konstrukcijos rūšis *(stereotype)* – tai vienas iš UML plėtros mechanizmų. Ji apibrėžia naują UML konstrukcijų klasę, kuria galima naudotis tame modelyje, kuriame ji apibrėžta.

UML savybė – tai su UML konstrukcija susieta reikšmė užduota meta-modelio atributu, asociacija arba žymėtąja reikšme.

UML žymė – tai „*vardas*", poroje "*vardas=reikšmė*". *(„tag")*

UML žymėtoji reikšmė – tai pora "*vardas=reikšmė*". Ją galima apibrėžti bet kuriai UML konstrukcijai. *(„tagged value")*

PS reikalavimas – tai sandoriu su užsakovu, specifikacija, standartu ar kitu juridinę galią turinčiu dokumentu numatyta sistemos savybė.

Vartotojo reikalavimai (operaciniai poreikiai) – tai reikalavimai, skirti būsimiems sistemos vartotojams bei jos užsakovams. *{ Turi būti suprantami žmonėms tik paviršutiniškai susipažinusiems kompiuteris ir PS }*
Sistemos reikalavimai – tai reikalavimai, rašomi kaip sudėtinė užsakovo ir vykdytojo sandorio dalis. *{ Tam tikru specialiu būdu struktūrizuotas detalus sistemos teikiamų paslaugų ir jos tenkinamų ribojimų aprašas. }*
Projektiniai reikalavimai – tai reikalavimai skirti vykdytojams. Susieja sistemos reikalavimus su jos realizacija. *{ Abstraktus PS įgyvendinimo būdo aprašas, naudojamas kaip išeities ribojimai detaliai projektuojant sistemą. }*
Funkciniai reikalavimai – Reikalavimai, nusakantys, kokias paslaugas privalo teikti sistema, kokia turi būti jos reakcija į konkrečius stimulus ir kaip ji turi elgtis konkrečiose situacijose.
Nefunkciniai reikalavimai – Ribojimai potencialiai galimų projektinių sprendimų aibei. *(pvz., maksimali trukmė, standartai, patikimumas)*
Reikalavimų formulavimas – tai procesas, kurio metu operaciniai poreikiai yra pertvarkomi į tikslų ir išsamų aprašą, ką sistema darys ir kaip ji veiks.

Gerai suformuluotas reikalavimas turi būti:

Abstraktus - jis suformuluotas vadovaujantis juodosios dėžės principu, t.y. Specifikuoja operacinę (stebimą iš išorės) sistemos savybę ir nieko nekalba apie tai, kaip tą savybę realizuoti sistemoje.

Išsamus - jis turi prasmę ne tik tuomet, kai yra nagrinėjamas kartu su kitais reikalavimais, bet ir tuomet, kai jis nagrinėjamas atskirai.

Tikslus - visi jame vartojami terminai turi griežtai apibrėžtas reikšmes.

Vienareikšmis - jo negalima interpretuoti (suprasti) keliais skirtingais būdais, t.y. igyvendinamas tik vienu būdu.

Verifikuojamas *(patikrinamas)* – jam yra žinomas ir prieinamas baigtinis, ir kainos bei kitais požiūriais priimtinas, procesas *(testavimas, stebėjimas, analizė ar kt.)*, kurį taikant galima nustatyti, ar reikalavimas tikrai yra įgyvendintas.

- Netikslūs reikalavimai yra neverifikuojami.

Įgyvendinamas – jam yra žinomas ir žinomas ir prieinamas toks*(ekonominiu, juridiniu bei kitais požiūriais)* priimtinas technologinis procesas, kurio inovaciniai slenksčiai gali būti pašalinti per priimtiną laikotarpį už priimtiną kainą ir kurį taikant galima sukurti sistemą, turinčią tuo reikalavimu specifikuojamą savybę.
Integruojamas – sujungus jį su kitais reikalavimais, yra gaunamas tarpusavyje suderintų reikalavimų rinkinys.
Lokalizuojamas – jį galima susieti su vienu ar keliais konkrečiais kuriamos sistemos komponentais, įgyvendinančiais tą reikalavimą.
Trasuojamas – jis yra vienareikšmiškai įvardinamas *(pvz., turi unikalų numerį)* ir turi nuorodą į savo šaltinį.
Unikalus – jame nėra kartojama kituose reikalavimuose pateikta informacija.
Glaustas – jame nėra pagrindimo, apibrėžčių ar kitų nebūtinų dalykų.
Suprantamas – jis parašytas nevartojant tik specialistams suprantamų terminų ir yra aiškiai pasakyta, kokią funkcinę ar nefunkcinę savybę privalo turėti sistema.

Duomenų filtravimas - tai procesas, kai iš įvesties ar išvesties duomenų srauto yra išrenkamas nurodytas sąlygas tenkinantis duomenų poaibis. *(pvz. INCLUDE employees IN report WHERE salary>5000)*
Kvalifikavimo išraiška – tai loginėmis operacijomis (IR, ARBA, NE ir t.t.) sujungti predikatai(sąlygos).
Sisteminis analitikas – tai esminė vykdytojų kolektyvo figūra, nustatantis projekto apimtis, tikslus, dažnai ir terminus.
Interviu – tai analitiko pokalbis su užsakovu, dalykinės srities specialistu, kuriamos sistemos būsimuoju vartotoju ar kitu asmeniu, kurio metu yra atsakoma į iš anksto parengto klausimyno klausimus ir visi atsakymai yra išsamiai dokumentuojami.

Interviu tipai:

- **įvadiniai interviu** – tikslas susipažinti su analizės objektu
- **pagrindiniai interviu** – tikslas surinkti informaciją reikalingą vartotojų poreikiams nustatyti ir sistemos reikalavimams suformuluoti

Darnos analizė – tai metodas kai tikrinamas ne analizės rezultatų korektiškumas, bet jų išsamumas, t.y. žiūrima, ar neliko "skylių".

Reikalavimų peržiūra - tai analizės rezultatų prezentacija specialistams, nedalyvavusiems atliekant analizę. Vienas iš efektyviausių vertinimo metodų.

Sistemos hierarchija – tai sistemos komponentų skaidymas į dalis pagal tam tikrą struktūrą.

PS architektūra – tai sistemos struktūriniai elementai, jų komponavimas į posistemius, pagal tam tikras taisykles *(architektūros stilių)*, bei bendras sistemos organizavimo būdas.

PS prototipas - tai sistemos modelis, naudojamas pačiai sistemai kurti arba vertinti.

Koordinavimo mechanizmai:

- **Tiesioginis valdymas** - kas nors vienas sprendžia, ką turi daryti visi kiti agentai. *(„Direct Supervision")*
- **Daugkartinių patikslinimų** - agentai, komunikuodami su kitais agentais, patys koordinuoja savo veiklą. *(„Mutual Adjustment")*
- **Darbo proceso, rezultatų ir darbo būdų standartizavimas** - visi agentai veikia pagal iš anksto nustatytas griežtas taisykles.

Funkcinis projektavimas - sistemą sudaro nuosekliai vienas po kito vykdomi moduliai, realizuojantys nepriklausomas funkcijas.

Funkcinis projektavimas - sistema traktuojama kaip funkcijų rinkinys. Sistema turi centralizuotą būseną, su kuria dirba visos funkcijos.

Objektinis projektavimas - Sistema traktuojama kaip sąveikaujančių objektų rinkinys. Sistemos būsena decentralizuota, kiekvienas objektas pats tvarko savo būseną. Objektai yra klasių egzemplioriai ir komunikuoja vienas su kitu keisdamiesi pranešimais.

Užduotimis grindžiamas projektavimas - sistemą sudaro lygiagrečiai vykdomi moduliai *(agentai)*, kiekvienas iš kurių vykdo kokias nors specifines užduotis *(jas moka vykdyti tik jis vienas)*.

Reikalavimų lokalizavimo rūšys:

- **Tiesioginis lokalizavimas** - lokalizuojami reikalavimai nekeičiami, o tik susiejami su konkrečiais komponentais.
- **Netiesioginis lokalizavimas** - lokalizuojami reikalavimai, reikalavimų nuleidimo žemyn metu, yra perrašomi (keičiami).

Reikalavimų trasavimas – tai savotiška reikalavimų inventorizacija, kuri padeda įsitikinti, kad reikalavimų lokalizavimas ir nuleidimas žemyn buvo atlikti teisingai.

Reikalavimų lokalizavimo matrica – tai reikalavimų ir komponentų lentelė, rodanti, koks reikalavimas lokalizuojamas į kokius komponentus.

Reikalavimų ryšio matrica -?

Reikalavimų trasavimo matrica - reikalavimų lentelė, rodanti, iš ko(kurio reikalavimo) reikalavimas išvestas ir kur jis lokalizuotas.

Reikalavimų nuleidimas žemyn – tai lokalizuotų reikalavimų perrašymas komponentų, kuriuose jie yra lokalizuoti, terminais.

Verifikavimo ir vertinimo metodai:

Peržiūra – tai vertinimo procedūra, skirta kokiai nors analizuojamai medžiagai išsiaiškinti ir perprasti.
Inspektavimas - tai verifikavimo procedūra,kuria siekiama atrasti darbo metu padarytas klaidas ir užtikrinti aukštą rezultatų kokybę.

Komponento rišlumas - tai suprojektuoto komponento "patvarumo" matas. Rišlus komponentas įgyvendina vieną sistemos funkciją (funkciniame projektavime) arba vieną sistemos esybę (objektiniame projektavime).

Komponentų sankiba – tai komponentų tarpusavio priklausomumo matas. Jei komponentai yra silpnai sukibę, tai padidėja tikimybė, kad, darant pakeitimus viename komponente, jie neturės jokio poveikio kitiems komponentams.

Projekto adaptuojamumas - kiek lengva projektą keisti pagal pasikeitusius poreikius.

- **Paveldėjimas** iš esmės pagerina **adaptuojamumą**.

PSGC – tai sistema (metodinė), skirta programų sistemoms kurti ir prižiūrėti.

Paskaitų klausimai-atsakymai

7 paskaita

Kas vadinama modeliu? (2)
Modelis – tai aiškiai nusakytą tikslinę paskirtį turintis supaprastintas sistemos, proceso, reiškinio ar kokio nors kito originalo analogas, tapatus tam originalui modeliavimo tikslų požiūriu.

Modelis pateikiamas forma, patogia naudotis modeliavimo tikslams.

Kaip užrašomi tvirtinimai ir teiginiai? (2)
Tvirtinimai ir teiginiai užrašomi kokia nors modeliavimo kalba (pvz., UML).

Kokiems tikslams naudojami modeliai programų sistemų inžinerijoje? (4)
PS inžinerijoje modeliai yra naudojami keliems tikslams:
1. Atskleisti ir tiksliai aprašyti verslo sistemų struktūrą ir elgseną (modeliuoti verslą tikslu atlikti vidinę analizę).
2. Tiksliai aprašyti būsimos PS reikalavimus ir jos struktūrą bei elgseną (įvairiais abstrakcijos lygmenimis) ir tuo pačiu sudaryti prielaidas visiems projekto dalyviams tiksliai ir vienareikšmiškai susitarti kokia ta sistema turėtų būti (reikalavimų modeliavimas, atliekant koncepcinį projektavimą).
3. Apgalvoti PS projektą.
4. Aprašyti projektavimo sprendimus.
5. Generuoti įvairius tarpinius darbo rezultatus.
6. Organizuoti, ieškoti, filtruoti, analizuoti ir redaguoti informaciją apie dideles sistemas.
7. Ekonomiškai vertinti skirtingus projektavimo sprendimus.
8. Kuriamai sistemai įgyvendinti.

Kiek lygmenų turi programų sistemų inžinerijoje naudojami modeliai? Kodėl? (4)

Priklausomai nuo modeliavimo tikslų modeliai gali įgyti skirtingus pavidalus ir būti pateikiami skirtingais abstrakcijos lygmenimis.

Ankstyvosiose projekto stadijose kuriami aukšto abstrakcijos lygmens modeliai padeda projekto dalyviams mąstyti kryptingai ir išryškina svarbiausias sistemos savybes. Tokie modeliai aprašo sistemos reikalavimus ir naudojami kaip išeities taškas PS projektavimui.

Laikui bėgant, modeliai evoliucionuoja. Konkretesni modeliai yra išvedami iš abstraktesnių. Laikui bėgant, pridedama vis daugiau ir daugiau detalių, atsiranda variantai:

- reikalavimų
- struktūrinio
- detali specifikacijos
- veikimo bei naudojimo pavyzdžių
- sistemos aspektų aprašų

Kas vaizduojama modeliuose? Semantika ir pateiktis. (2)

Modeliai turi du aspektus:
1. ***Semantinė informacija*** *(semantika)*
Semantinis modelio aspektas aprašo PS kaip loginių konstrukcijų (klasių, asociacijų, būsenų, pranešimų, užduočių) tinklą. Dažnai semantinė informacija tapatinama su pačiu modeliu.
Semantinis modelis turi sintaksinę struktūrą, taisykles, nusakančias, kokie modeliai yra sintaksiškai teisingi ir vykdymo dinamiką. Kiekvienas iš šių aspektų dažniausiai yra aprašomas atskirai, tačiau visi jie yra persipynę vienas su kitu ir yra to paties modelio dalys.
2.***Vizualinė pateiktis*** *(notacija)*
Vizualizuojant modelį, semantinė informacija pateikiama pavidalu, pritaikytu peržiūrinėti ir redaguoti tą informaciją.
Vizualizuojant, modelis pateikiamas žmogui lengvai suprantama forma.
Jokia papildoma informacija nėra pridedama, tačiau esama informacija organizuojama kitaip, negu ji yra

saugoma kompiuteryje (ten ji saugoma forma, patogia apdoroti tą informaciją kompiuteriu).
Tinkama parinkta notacija įgalina lengvai skaityti modelį!

Išvardinkite UML™ diagramas ir trumpai apibūdinkite kiekvienos diagramos paskirtį. (4)
Elgsenos diagramos:

- ***Užduočių diagramos*** - modeliuoja globalų *(stambaus plano)* požiūrį į sistemą, apibūdina sąveiką su sistema per jos interfeisus, aprašo sistemos funkcionalumą kaip diskrečių užduočių *(verslo transakcijų)* rinkinį.
- ***Sąveikos diagramos*** *(Sekų diagrama, Ansamblių diagrama)* - aprašo, kaip užduotys realizuojamos per objektų tarpusavio sąveiką

Statinės struktūros diagramos:

- ***Klasių diagrama*** - aprašo sistemos tipų struktūrą
- ***Objektų diagrama***
- ***Paketų diagrama***

Dinaminio modeliavimo diagramos:

- ***Būsenų diagrama*** - aprašo individualių objektų elgseną, modeliuoja sistemos dinaminę elgseną
- ***Veiklos diagrama*** - būsenų diagramų porūšis, parodantis veiklos būsenų (būsenų, kuriose kas nors yra padaroma) sekas.

Realizavimo diagramos:

- ***Komponentų diagrama*** - parodo iš kokių fizinių komponentų yra sudaryta sistema
- ***Išdėstymo diagrama*** - parodo, kaip kompiuterių tinkluose fiziškai išdėstomi sukurtos sistemos komponentai

Kokiems tikslams naudojamos UML™ diagramos kuriant programų sistemas? (2)
UML diagramos naudojamos trims skirtingiems tikslams:

- ***Koncepciniam modeliavimui:*** modeliuoti realaus pasaulio sistemas (pvz., verslo sistemas);
- ***Kuriamoms sistemoms*** *(pvz., PS)* ***specifikuoti***, jas projektuoti koncepciniu ir architektūriniu lygmenimis;
- ***Kuriamoms sistemoms realizuoti:*** projektuoti eskiziniu ir detaliuoju lygmenimis.

Išvardinkite užduočių diagramos elementus ir paaiškinkite kaip jie vaizduojami. (5)
Užduočių diagramos elementai:

- Sistema
- Užduotys – užduotys, požduotys, plėtiniai, išplėtimo taškai
- Agentai *(aktoriai)*
- Ribojimai *(taisyklės, aprašančios ką galima ir ko negalima daryti. Rašomi skliaustuose {}. Galima naudoti OCL (object constraints language). Aprašo predikatus, kurie turi būti teisingi.)*
- Asociacijos *(jungia agentus su jų vykdomomis užduotimis)*
    - sąveika
    - apibendrinimas *(rodo, kad viena užduotis yra panaši į kitą, bet yra šiek tiek bendresnė)*
    - pastabos ryšys
- Priklausomybės (*parodo, kad vieno elemento apibrėžties pokyčiai iššaukia kito elemento apibrėžties pokyčius)*
    - include (use) (rodo, kad viena užduotis naudojasi kita užduotimi kaip požduotimi)
    - extend (rodo, kad viena užduotis praplečia kitą. Labiau formalizuota už apibrendrinimą)
- Paketai *(koncepcinio lygmens konstrukcija; naudojamas projektavimo metu. Grupuoja modelio elementus)*
- Pastabos *(galima įrašyti bet kokį tekstą, su objektų susiejamos pastabos ryšiu. Šiuo ryšiu galima susieti ir ribojimus)*
- Tekstiniai paaiškinimai *(neprivalomi)*

Kokia užduočių diagramų paskirtis? (3)
Užduočių diagrama pateikia išorinį sistemos vaizdą.

Kaip reikia patikslinti užduočių diagramas, aprašant kas vyksta vykdant užduotį? (2)
Užduočių diagramų prasmė patikslinama kitomis elgsenos diagramomis, aprašančiomis užduočių vykdymą.
tikriausiai nepilnas

Koks sistemos aspektas aprašomas užduočių diagramomis ? (1)
Užduočių diagramomis modeliuojamas globalus (stambaus plano) požiūris į sistemą, paprastai jis vadinamas išoriniu požiūriu.

Ką užduočių diagramose vaizduoja užduotis? (1)
UML konstrukcija užduotis *(use case)* aprašo kokį nors bendrą sistemos funkcionalumą *(verslo transakcijas, procesus)*.

Kas užduočių diagramose gali būti vaizduojama kaip agentas? (1)
Agentas yra bet kas, kas yra sistemos išorėje ir kartu su sistema dalyvauja užduoties vykdyme. Tai gali būti vartotojas arba kita sistema. Agentas privalo turėti vardą.

Kokiems tikslams užduočių diagramose agentas naudoja užduotį? (1)
Agentas “naudoja" užduotis savo tikslams pasiekti. Pagal tai, kokias užduotis agentas atlieka, nusakomas jo vaidmuo sistemoje

Kaip užduočių diagramose aprašomas agento vaidmuo sistemoje? (1)
Agentas su užduotimi jungiamas sąveikos (interaction) asociacija

Kokias asociacijų rūšis galima naudoti užduočių diagramose? (2)
Sąveikos asociacija
Apibendrinimo asociacija
Pastabos ryšio asociacija

Kas užduočių diagramose vaizduojama «include» ryšiu? (1)
«include» ryšys aprašo vienos ar kelių užduočių naudojamas použdutis.

Kas užduočių diagramose vaizduojama «extend» ryšiu? Kas vadinama išplėtimo tašku? (3)
«extend» ryšys aprašo neprivalomas (vykstančias prie tam tikrų sąlygų) použdutis.

Kas užduočių diagramose vaizduojama apibendrinimo ryšiu?(1)
Apibendrinimo asociacija – tai neformalus būdas pavaizduoti, kad viena užduotis yra panaši į kitą, bet, funkcionalumo požiūriu, yra už ją šiek tiek bendresnė.

Konkretesnė užduotis paveldi bendresnės užduoties funkcionalumą ir jį kuo nors šiek tiek papildo.

Išvardinkite sekų diagramos elementus ir paaiškinkite kaip jie vaizduojami. (6)
- Vaidmenys - agentai *(actors)*, objektai
- Pranešimai – sinchroniniai, asinchroniniai, konstruktoriai, destruktoriai, rekursyvūs, signalai, return
- Gyvavimo atkarpos *(lifelines)*
- Aktyvumo stulpeliai *(activity boxes)*
- Išsišakojimai *(forks)*
- Jungimosi taškai *(joins)*
- Šakos *(branches)*
- Valdymo gijos *(threads of control)*
- Ribojimai - laiko
- Paketai
- Pastabos
- Tekstiniai paaiškinimai *(neprivalomi)*

Kokia yra sekų diagramų paskirtis? (4)
Sekų diagramos aprašo objektų komunikavimą laiko tėkmėje, atliekamą keičiantis tarpusavyje pranešimais. Sekų diagrama paprastai vaizduoja kokios nors užduoties įvykių srautą.
Kas sekų diagramose vadinama gyvavimo atkarpa? (1)

Objektų rolės vaizduojamos gyvavimo atkarpomis:

Brūkšnine linija vaizduojami objekto pasyvaus egzistavimo periodai Aktyvaus egzistavimo (veikimo) periodai vaizduojami aktyvumo stulpeliais (siaurais stačiakampiais).

Kiekvienas aktyvus objektas turi savo nuosavą valdymo giją, vykdomą lygiagrečiai su kitų aktyvių objektų valdymo gijomis

Kaip sekų diagramose yra modeliuojami objektai? (1)

Sekų diagramoje po kiekvienu objektu vaizduojama jo gyvavimo atkarpa, parodanti objekto aktyvumo periodus.

Kas vadinama pranešimu? Kaip sekų diagramose yra modeliuojami pranešimai? Kokias pranešimų rūšis galima modeliuoti sekų diagramose? (3)

Objektai komunikuoja tarpusavyje keisdamiesi pranešimais;

Pranešimai išdėstomi iš viršaus žemyn ta tvarka, kuria jie yra siunčiami;

- Return pranešimas
- Rekursyvusis pranešimas
- Asinchroninis pranešimas
- Sinchroninis pranešimas

pranešimas gali būti signalas (t.y. išreikštinė, įvardinta, asinchroninė objektų sąveika) arba call tipo pranešimas, t.y. sinchroninis pranešimas.

Kaip sekų diagramose yra modeliuojami konstruktoriai ir destruktoriai? (2)

Naujo objekto kūrimas (konstruktorius) modeliuojamas kaip kuriančiojo objekto generuojamas įvykis, adresuotas kuriamojo objekto klasei (o ne objektui, kaip kiti pranešimai).

{apie destruktorius nėra}

Kaip sekų diagramose yra modeliuojamas šakojimasis? (2)
Išsišakojimą galima modeliuoti pranešimus suskirstant į nuoseklaus vykdymo valdymo gijas.
Pačios gijos gali būti vykdomos lygiagrečiai.
Gijos sinchronizuojamos ribojimais užduodamais ant pranešimų parametrų.

Kas vadinama ansambliu? (1)

Ansamblį sudaro ansamblio dalyviai ir tik to ansamblio kontekste turintys aiškiai apibrėžtą prasmę tuos dalyvius siejantys sąryšiai.

Išvardinkite ansamblių diagramos elementus ir paaiškinkite kaip jie vaizduojami.. (6)

- Vaidmenys – Agentai, objektai, aktyvūs objektai, multiobjektai
- Pranešimai
- Sąryšiai
- Reikalavimai
- Paketai
- Pastabos

Kokia yra ansamblių diagramų('comunication diagram') paskirtis? (4)
Kaip ir sekų diagramose, jose parodoma, kaip objektai komunikuoja laiko tėkmėje.
Tačiau, skirtingai negu sekų diagramose, laikas parodomas ne specialia ašimi, bet pranešimų numeravimo tvarka.
Jose parodomi objektų tarpusavio sąryšiai, bet pranešimų išsidėstymas laike matosi blogiau.

Kas vadinama pranešimu? Kaip ansamblių diagramose yra modeliuojami pranešimai? Kokias pranešimų rūšis galima modeliuoti ansamblių diagramose? (3)
Objektai komunikuoja tarpusavyje keisdamiesi pranešimais. Return, sinchroniniai, asinchroniniai, rekursyvūs pranešimai, signalai. Siunčiant pranešimą valdymo srautas perduodamas gavėjui. Ansamblių diagramose perduodami pagal sąryšius, numeruojami jų perdavimo eilės tvarkai parodyti.

Kaip ansamblių diagramose yra modeliuojami konstruktoriai ir destruktoriai? (2)
Vykdymo metu sukuriamiems objektams nurodoma savybė {new};
Vykdymo metu sunaikinamiems objektams nurodoma savybė {destroyed};
Vykdymo metu sukuriamiems ir sunaikinamiems objektams nurodoma savybė {transient}.
Kas vadinama multiobjektu? Kaip multiobjektai yra modeliuojami ansamblių diagramose? (2)
Multiobjektai - vaizduoja asociacijos “daugelis” gale esančias objektų aibes.
Jie naudojami parodyti operacijas, atliekamas ne su paskirais objektais, bet su visa objektų aibe.
Tai tas pats, kaip asociacija su kardinalumu “daug”, naudojama prieigai prie ja siejamų objektų.

Kuo skiriasi aktyvieji ir pasyvieji objektai? Kaip modeliuojami aktyvieji objektai sekų diagramose ir kaip jie modeliuojami ansamblių diagramose? (3)
Aktyviuoju vadinamas objektas, kuriam perduota valdymo gija ir kuris gali inicijuoti valdymo veiklas.
Pasyviuoju vadinamas objektas, disponuojantis duomenimis, bet negali inicijuoti valdymo veiklų.
Ansamblių diagramose aktyvieji objektai vaizduojami paprastų objektų žymenimis, pastorinant jų linijas.

Kaip ansamblių diagramomis yra aprašomi tipiniai projektavimo sprendimai? (3)

A collaboration can be used to specify the implementation of design constructs. For this purpose it is necessary to specify its context and interactions. It is also possible to view a collaboration as a single entity from the outside. For example, this could be used to identify the presence of design patterns within a system design. A pattern is a parameterized collaboration; in each use of the pattern, actual classes are substituted for the parameters in the pattern definition.

Note that patterns as defined in Design Patterns by Gamma, Helm, Johnson, and Vlissides include much more than structural descriptions. UML describes the structural aspects and some behavioral aspects of design patterns, but UML notation does not include other important aspects of patterns, such as usage trade-offs or examples. These must be expressed in text or tables.

Išvardinkite klasių diagramos elementus ir paaiškinkite kaip jie vaizduojami. (7)
Klasės, objektai, interfeisai, metaklasės, abstrakčiosios klasės, šablonai, utilitos, paketai, pastabos, ribojimai, priklausomybės ir asociacijos – apibendrinimas, klasifikavimas, kompozicija, agregavimas.
Kokia yra klasių diagramų paskirtis? (4)
Klasių diagramomis aprašomi sistemoje leistinų objektų tipai ir juos siejantys statiniai ryšiai.

Klasė turi tam tikrą verslo funkcionalumą, vadinamą metodais (operacijomis), ir unikalias, tik jai būdingas struktūrines savybes, vadinamas atributais.

Kaip klasių diagramose modeliuojamos klasės? (4)
Nėra atsakymo

Kaip nustatyti, kokios klasės turi būti modeliuojamos klasių diagrama? (2)
Kokios klasės yra reikalingos, nustatoma nagrinėjant sekų diagramose pavaizduotus objektus. Pavadinimai klasėms suteikiami naudojantis dalykinės srities terminų žodynu (jis yra vadinamas ontologija) arba pasirinktąja dalykinės srities metafora.

Kaip nustatyti, kokius atributus turi turėti modeliuojama klasė? (2)
Atributai nustatomi nagrinėjant pranešimų parametrus ir “prieš” bei “po” sąlygas sekų diagramose.

Kaip nustatyti, kokias operacijas turi turėti modeliuojama klasė? (2)
Operacijas galima nustatyti nagrinėjant sekų diagramas.

Kaip nurodomas klasės elementų matomumas? Kokios yra klasės elementų matomumo rūšys? (2)
matomumas vardas: tipas = standartinė reikšmė
matomumas pavadinimas (parametrų sąrašas) : rezultato tipas
public (+), private (-), protected (#) ar package (~).
dar trūksta apie tai, kokie paveldėti matomi elementai
Kas vadinama statiniais nariais? Kaip jie modeliuojami klasių diagramose? (2)

Statiniais klasės nariais (pačios klasės savybėmis) galima naudotis net ir tuomet, kuomet klasė yra tuščia.

Klasės apraše statiniai nariai yra pabraukiami.

Kas vadinama parametrizuotomis klasėmis (šablonais)? Kam jų reikia? Kaip jos modeliuojamos klasių diagramose? (4)
Tipizuotose programavimo kalbose šablonai dažniausiai yra naudojami skirtingų tipų objektų rinkiniams aprašyti.

Koncepciniame modeliavime be jų galima ir apseiti, nes čia objektų rinkinius galima modeliuoti ir kitais būdais.

Eskizinio ir detalaus projektavimo metu šablonus rekomenduojama naudoti tuomet, kuomet juos palaiko pasirinktoji programavimo kalba, pvz., C++.

Kuo skiriasi klasė ir tipas? (3)
Tipas:
- protokolas, kurį supranta objektas;
- leistinų operacijų rinkinys.
Klasė:
- realizavimo konstrukcija;
- realizuoja vieną ar kelis tipus.

Kam reikia abstrakčių klasių? Kaip jos modeliuojamos klasių diagramose? (3)
Nėra atsakymo

Kas vadinama asociacijomis? Kam jų reikia? Kaip jos modeliuojamos klasių diagramose? (4)
Asociacija parodo, kad dvi klases sieja tam tikras ryšys. Asociacijos turi du galus. Jie vadinami asociacijos *vaidmenimis*. Kiekvienas vaidmuo turi pavadinimą, *kardinalumą*, *navigavimo kryptį* ir *tipą*.

- **Kardinalumas** nurodo, kiek tame vaidmenyje esančių objektų gali būti susieti asociacijos generuojamu sąryšiu. Pvz: Mokyklą lanko *daug* Studentų (sąryšis daug su vienu, *:1). Pvz2: Bet kuris Studentas lanko *tik vieną* Mokyklą (1:*).
- **Navigavimo kryptis**: kartais vaidmeniui yra dedama rodyklė, nurodanti navigavimo kryptį. Ji nustato, kuri iš klasių yra atsakinga už ryšio tarp klasių palaikymą. Pvz: klasė Mokykla yra *atsakinga už žinojimą*, kurie Studentai ją lanko.
- **Tipas**: Vaidmenys priklauso vienam iš trijų tipų: asociacija, kompozicija, agregavimas.

Kuo skiriasi agregavimas ir kompozicija? Kokia šių asociacija paskirtis? Kaip jos modeliuojamos klasių diagramose? (4)

Naikinant kompoziciją, yra sunaikinamos ir visos jos dalys. Vienu metu objektas gali priklausyti tik vienai kompozicijai.

Agregatuose tas pats objektas gali iš karto priklausyti keliems agregatams. Pavyzdžiui, ta pati siena tuo pat metu gali būti kelių kambarių dalis. Sunaikinus agregatą, dalis gali išlikti.

Kokia šių kvalifikuotomis asociacijų paskirtis? Kaip jos modeliuojamos klasių diagramose? (4)

Atributų sekcijoje įrašomi vienas ar daugiau atributų, kuriuos galima panaudoti kaip indeksus pereinant asociacija iš kvalifikuotos klasės į tikslinę klasę, esančią kitame asociacijos gale.

Kokia asociacijos klasių paskirtis? Kaip jos modeliuojamos klasių diagramose? (4)
Ji nusako, kokias savybes turi asociacija.

Paaiškinkite kuo skiriasi asociacijos ir operacijos. (3)
Asociacijos modeliuoja nuolat tarp objektų egzistuojančius sąryšius.

Operacijos sąryšius tarp objektų kuria ir naikina dinamiškai.

Ką paveldi klasės poklasis? Kokiems tikslams naudojami apibendrinimo ryšiai? Kaip jie modeliuojami klasių diagramose? (4)
Klasės poklasis paveldi visas klasės savybes - atributus, operacijas, agregavimo bei kompozicijos ryšius, kitas asociacijas.
Kaip yra modeliuojamos priklausomybės klasių diagramose? (4
Pavyzdys: klientą ir tiekėją sieja priklausomybė, jei klientas apie tiekėją nieko nežino. Tai naudojimo ryšys, parodantis, kad kokio nors daikto specifikacijos pokyčiai padaro poveikį kitam, jį naudojančiam, daiktui

Išvardinkite priklausomybių rūšis ir paaiškinkite kokiems tikslams naudojama kiekviena iš priklausomybių rūšių. (7)
«trace»: skirtingų modelių elementų koncepcinė priklausomybė.
«refinement»: priklausomybė tarp dviejų skirtingų to paties koncepto versijų, pavyzdžiui, apibrėžtų skirtinguose abstrakcijos lygmenyse ar skirtingose projekto stadijose.
«realisation»: specifikaciją ir jos realizaciją siejanti priklausomybė.
«trace», ***«refinement»***, ***«realisation»*** ir ***«derivation»*** priklausomybės yra abstrakcijos priklausomybės; jos sieja dvi to pačio dalyko versijas.
«derivation»: elementą ir iš jo išskaičiuotą (išvestą) elementą siejanti priklausomybė.

«usage»: teiginys, kad vieno elemento elgsena ar realizavimo būdas veikia kito elemento elgseną ar realizavimo būdą.
«access»: leidimas vienam paketui prieiti prie kito paketo turinio.
«binding»: reikšmių priskyrimas parametrams.
«call»: viena klasė kviečia kitos klasės operaciją.
«friend»: leidimas elementui prieiti prie kito elemento, nepriklausomai nuo jo matomumo, turinio.
«import»: leidimas paketui prieiti prie kito paketo turinio ir pridėti to paketo vardų aliasus į savo vardų erdvę.
«instantiation»: vienos klasės metodas kuria kitos klasės objektus.
«send»: signalo siuntėją ir jo gavėją siejanti priklausomybė.
«instance of»: parodo, kad vienas elementas yra kito elemento egzempliorius. Griežtai kalbant, tai yra ne priklausomybė, bet metaryšys.
«parameter»: operaciją ir jos parametrus siejanti priklausomybė.

Kokiems tikslams yra naudojamos realizavimo priklausomybės? Kaip jos modeliuojamos? (4)
Naudojama susiesti specifikacijai su jos realizacija

Kam UML kalboje reikalingi interfeisai? Kaip jie modeliuojami? (4)
Interfeisą gali realizuoti viena ar kelios klasės ir kiekviena iš jų privalo realizuoti interfeise specifikuotas operacijas.

Interfeisai neturi iš jų išeinančių asociacijų.

Kam UML kalboje reikia utilitų? Kaip jos modeliuojamos klasių diagramose? (3)
Tai ne realaus pasaulio modeliavimo, bet techninė konstrukcija, įvesta tik dėl patogumo.
Ji žymima klasės žymeniu, nurodant rūšį ***«utility»***.

Kam UML kalboje reikia metaklasės? Kaip jos modeliuojamos klasių diagramose?(3)
Metaklasės žymimos klasės žymeniu, nurodant rūšį ***«metaclass»***

Kam UML kalboje naudojami objektu? Kaip jie modeliuojami? (3)
Nėra atsakymo.

Paaiškinkite kuo skiriasi klasių ir objektų diagramos. (3)
Objektu vardai pabraukti, su pavadinimu po dvitaškio: studentas:petras

Kam reikalingi sudėtiniai objektai? Kaip jie modeliuojami? (3)
Sudėtiniai objektai yra sudėtinių klasių realizacijos.
Jie primena ansamblius, bet vaizduojami kaip statiniai modeliai ir modeliuojami panaudojant kompozicijas.

Išvardinkite būsenų diagramos elementus ir paaiškinkite kaip jie vaizduojami. (6)
Būsenos – pradinė, galinė, sudėtinė *(nested)*
Perėjimai *(transitions)* – vidiniai, sudėtiniai, nupjauti *(stubbed)*
Valdančiosios piktogramos - signalo siuntimas, signalo gavimas, atidėtas įvykis
Ribojimai - perėjimo sąlyga, when, after
Veiksmai – enter, exit, do, on, send aprašai
Pastabos
Įvykiai - keičiantys įvykiai, signalai, kvietimai *(call)*, laiko įvykiai
Šakojimai - valdymo iššakojimas,lygiagrečios gijos, sinchronizacija
Kokia yra būsenų diagramų paskirtis? (4)

Nusako sistemos elgseną aprašant, kokias vidinių operacijų sekas iššaukia išoriniai stimulai

Parodo:
- Visus tai klasei darančius poveikį įvykius.
- Visas galimas tos klasės objektų būsenas.

Naudojama:
- Kryžminiam sekų diagramos tikrinimui.
- Operacijų trigeriams/parametrams/ribojimams nustatyti.
- Sistemos interfeisui projektuoti.

Kaip nustatyti, kokias būsenas turi objektas? Kaip modeliuojamos būsenos? (6)
Būsena gali būti susieta su veikla, aprašančia vykdomą funkciją. Būsena vaizduoja objekto gyvavimo periodui priklausantį laiko intervalą, kurio metu objektas tenkina tam tikras sąlygas, vykdo tam tikrus veiksmus ar laukia kol įvyks koks nors įvykis

Pradinė yra būsena, kurioje prasideda modeliuojama objekto elgsena.

- Klasės būsenų diagrama privalo turėti vieną ir tik vieną pradinę būseną.
- Joje neleidžiami įvykių trigeriai.
- Joje gali būti duota šakojimosi sąlyga.
- Objektai negali užsilaikyti pradinėje būsenoje.

Pabaigos būsenoje pasibaigia elgseną modeliuojančios būsenų mašinos veikimas. Klasės būsenų diagrama gali turėti kelias pabaigos būsenas.

Būsena žymima stačiakampiu su užapvalintais kampais. Žymuo turi vieną arba daugiau sekcijų: pavadinimo ir vidinio perėjimo.

Paaiškinkite, kas vadinama perėjimu, perėjimo sąlyga ir perėjimo veiksmu. (4)
Tokiais perėjimais yra modeliuojami įvykiai, kuriems įvykus objekto būsena nepakinta.

Įėjimas į būseną (išskyrus grįžimą iš žemesnio lygmens būsenų) ir išėjimas iš būsenos (išskyrus atvejus, kai einama į žemesnio lygmens būsenas) yra vaizduojami kaip vidiniai perėjimai, žymimi tarnybiniais žodeliais "*entry*" ir "*exit*" atitinkamai, tačiau tai nėra tikri vidiniai perėjimai.

Kas vadinama sudėtiniu perėjimu? Kokia tokių perėjimų paskirtis? Kaip jie modeliuojami būsenų diagramose? (5)
Sudėtinis perėjimas - turi keletą pradinių ir/arba keletą tikslinių būsenų. Tokie perėjimai naudojami valdymą išskaidyti į kelias gijas (nepanaudojant lygiagrečių būsenų) ir/arba tokioms gijoms sinchronizuoti.

Kas vadinama nukirstuoju perėjimu? Kokia tokių perėjimų paskirtis? Kaip jie modeliuojami būsenų diagramose? (5)
(Stubbed transitions) Įdėtinės būsenos gali būti suspaustos. Perėjimai į įdėtinę būseną yra vaizduojamas kaip perėjimas į pačią detaliausią iš matomų suspaustosios būsenos būsenų. Toks perėjimas, jeigu jis neišeina iš neturinčios žymės galinės būsenos ir neįeina į neturinčią žymės pradinę būseną, gali (bet neprivalo) būti pavaizduotas kaip išeinantis iš nuokirtos *(stub)* arba į ją įeinantis. Nuokirtos vaizduojamos mažu vertikaliu brūkšneliu, apimančiosios būsenos viduje. Nuokirta parodo, kad perėjimas eina į suspaustąją būseną arba iš jos. Perėjimas iš nuokirtos į pradinę būseną ir perėjimas iš galinės būsenos į nuokirtą yra neleistini.

Kokia vidiniu perėjimų paskirtis? Kaip jie modeliuojami būsenų diagramose? (5)

Įėjimas į būseną *(išskyrus grįžimą iš žemesnio lygmens būsenų)* ir išėjimas iš būsenos *(išskyrus atvejus, kai einama į žemesnio lygmens būsenas)* yra vaizduojami kaip vidiniai perėjimai, žymimi tarnybiniais žodeliais "entry" ir "exit" atitinkamai, tačiau tai nėra tikri vidiniai perėjimai.

Perėjimas iš būsenos atgal į tą pačią būseną *(perėjimas į save)* taip pat nėra vidinis perėjimas

Kas vadinama sudėtine (įdėtine) būsena? Kokia tokių perėjimų paskirtis? Kaip jos modeliuojamos būsenų diagramose? (6)
Diagramoms supaprastinti, UML numato sudėtines *(įdedamas viena į kitą)* būsenas. Šis mechanizmas leidžia pavaizduoti būseną skirtingais abstrakcijos lygmenimis.

Būsena gali būti dekomponuota į nuoseklias (IR) arba alternatyvias (ARBA) būsenas.

Kokiems tikslams įvykiai naudojami modeliuose? Kaip jie modeliuojami būsenų diagramose? (6)
Trūksta atsakymo

Kaip būsenų diagramose yra modeliuojamas šakojimasis? (2)
Perėjimas gali būti padarytas sudėtingesniu, įvedant sąlygos žymenų medį. Tai ekvivalentu paprastų perėjimų rinkiniui (jį sudaro to medžio atkarpos), kuriame kiekvienas perėjimas pažymėtas visų sąlygų konjunkcija.

Kas vadinama būsenos istorijos indikatoriumi? Kam jų reikia? Kaip jie modeliuojami būsenų diagramose? (4)
Perėjus į indikatorių vedančiu perėjimu, yra atkuriama indikatoriaus kompleksinėje srityje *(t.y. įskaitant visas būsenas, iš kurių galima pereiti į indikatorių)* objekto turėta paskutinė būsena; pereinant į atkurtą būseną, yra atliekami visi veiksmai, kuriuos reikia atlikti pereinant į tą būseną. Indikatorius 'H' atkuria to paties lygmens būsenas.

Greta paviršinio būsenos istorijos indikatoriaus galima naudotis ir giluminiu būsenos istorijos indikatoriumi. Jis žymimas 'H*'. Jis atkuria indikatoriaus kompleksinėje srityje objekto turėtą paskutinę bet kurio lygmens būseną.

Būsenos srityje gali būti abu, paviršinis ir giluminis, istorijos indikatoriai.

Indikatoriaus galiojimo sritis yra būsenos sritis, kurioje jis pavaizduotas. Į indikatorių gali būti pereinama pagal perėjimus iš bet kurio būsenų skaičiaus. Iš indikatoriaus gali išeiti ne daugiau kaip vienas žymės neturintis perėjimas; jei į būsenos sritį niekuomet nebuvo įeita, šis perėjimas identifikuoja nutylimą "ankstesnę būseną".

Kaip būsenų diagramose yra modeliuojami konstruktoriai ir destruktoriai? (4)
Trūksta atsakymo

Išvardinkite veiklos diagramos elementus ir paaiškinkite kaip jie vaizduojami. (5)
- Būsenos
- pradinė *(juodas rutuliukas)*
- veiksmo *(stačiakampis apvaliais kampais)*
- galinė *(juodas rutuliukas apvestas juoda linija)*
- Šakojimas *(branching)* - nuoseklus šakojimas*(sprendimo priėmimas),* iššakojimas*(fork),* sujungimas*(join),* suliejimas *(merge)*
- Atsakomybės juostos *(swimlines)*
- Veiklos objektai *(kvadratas)* - už veiksmą atsakingas objektas, objektų srautas, objekto būsena
- Perėjimai *(rodyklės)*
- Paketai
- Pastabos *(tekstas su linija jungianti komentuojama objekta, burbuliukas linijos gale)*
- Valdančiosios piktogramos - signalo siuntimas, signalo gavimas, atidėtas įvykis, laiko įvykis

Kokia yra veiklos diagramų paskirtis? (4)
- Apskritai, veiklos diagrama yra Petri tinklas, tačiau dažniausiai tai yra paprastesnis atvejis, būtent, baigtinė būsenų mašina.
- Šiose diagramose dėmesys sutelkiamas į valdymo ir objektų srautus, kuriamus atliekant vidinį apdorojimą (o ne išorinių įvykių pasekmėje, kaip būsenų diagramose).
- Diagramoje galima parodyti lygiagrečias bei nuo sąlygų priklausančias elgsenas.
- Dažniausiai veiklos diagrama aprašo kokios nors klasės objektų kitimą (pvz., juos apdorojant verslo sistemoje) arba kokios nors užduoties ar operacijos realizavimo būdą.

Kas vadinama veiksmo būsena? Kokiems tikslams naudojamos tokios būsenos? Kaip jos modeliuojamos veiklos diagramose? (3)
Terminas veiksmo būsena yra trumpinys, kuriuo apibūdinamos būsenos, slepiančios savyje kokį nors vidinį veiksmą ir turinčios bent vieną perėjimą į kitą būseną, inicijuojamą išreikštiniu būdu nespecifikuoto vidinio veiksmo baigimo įvykio (jei perėjimai yra sąlyginiai, jų gali būti keli). Veiksmo būsenos žymimos stačiakampiu su užapvalintais galais. Veiksmas aprašomas žymens viduje

- Nereikalaujama, kad veiksmas diagramoje būtų unikalus.

Kam veiklos diagramose yra reikalingi sprendimų priėmimai ir kaip jie yra modeliuojami? (3)
Jei perėjimai priklauso nuo ant apdorojamo objekto užduotų predikatų, diagramoje turi būti numatytas sprendimo, kurį perėjimą parinkti, priėmimas.

- Sprendimo priėmimą galima pavaizduoti pažymint perėjimus atitinkamomis *sąlygomis (kaip būsenų diagramose).
- Sprendimo priėmimą taip pat galima pavaizduoti rombu, į kurį įeina vienas ar daugiau lankų ir iš kurio išeina du ar daugiau lankų, pažymėtų atitinkamomis sąlygomis, nepriklausančiomis nuo išorinių įvykių.
- Atitinkamos šakos pabaiga taip pat gali būti pažymėta rombu.

Kaip veiklos diagramose yra modeliuojamas šakojimasis? (2)
Iššakojimas parodo, kad paprastas perėjimas į būseną iššaukia kelis perėjimus iš tos būsenos. Jie vyksta lygiagrečiai. Jei veiklos turi būti sinchronizuotos, reikalingas sujungimas sujungiantis iššakojimo išskaidytą valdymo srautą. Iššakojimas reikalingas tuomet, kuomet paprastas perėjimas išsišakoja į kelias šakas. Šakos turi būti pažymėtos nepersikertančiomis sąlygomis, leidžiančios parinkti vieną ir tiktai vieną iš jų.

Kokia atsakomybės juostų paskirtis? Kaip jos modeliuojamos veiklos diagramose? (3)
Veiksmai gali būti grupuojami į atsakomybės juostas.

- Atsakomybės juostos verslo modelyje dažniausiai sutampa su organizaciniais padaliniais.
- Veiklos diagrama gali būti padalinta į "atsakomybės juostas", juostos viena nuo kitos skiriamos ištisinėmis vertikaliomis linijomis.
- Kiekviena atsakomybės juosta apima atsakomybę už dalį diagrama aprašomų veiklų ir gali būti įgyvendinta vienu ar keliais objektais.

Kas tai yra veiksmų ir objektų srautai? Kaip jie modeliuojami veiklos diagramose? (4)
Veiklas realizuoja objektai, jos manipuliuoja objektais.
Yra dvi objektų rūšys:

- Objektai, atsakingi už veiksmo vykdymą.
- Objektai, kurių reikšmes veiksmas naudoja arba sukuria.
  - Pastarieji modeliuojami pranešimais siuntinėjamais tarp objekto, kurio pokyčius modeliuoja diagrama, ir objektų, kurie yra naudojami kaip diagramoje modeliuojamų veiksmų įeiga ar išeiga.

Objektai *(vaizduojami stačiakampiais)*, pereina iš veiksmo į veiksmą punktyrinių rodyklių pagalba. Tas pats objektas dažniausiai naudojamas kaip vieno veiksmo išeiga ir vieno ar daugiau veiksmų įeiga.

Kas tai yra valdančiosios piktogramos? Kokiems tikslams jų reikia? Kaip jos naudojamos veiklos diagramose? (6)
Panaudojant valdančiąsias piktogramas, galima užduoti tam tikrą su perėjimais siejamą informaciją.

- ***Signalo gavimas:*** Žymimas įgaubtuoju penkiakampiu, gautu trikampį iškerpant iš stačiakampio. Signalo signatūra rašoma penkiakampio viduje.(pacman.. tik kampuota galva.. :))
- ***Signalo siuntimas:*** Žymimas iškiliuoju penkiakampiu, gautu sujungus stačiakampį ir trikampį. Signalo signatūra rašoma penkiakampio viduje.
- ***Laiko įvykiai:*** Tai kalendoriniai įvykiai, įvykus kuriems turi būti pradėti vykdyti koks nors veiksmas ar kokia nors veikla.vazduojamas dviem trikampiais (smelio laikrodzio pavidalas.)

Išvardinkite komponentų diagramos elementus ir paaiškinkite kaip jie vaizduojami. (5)

- Komponentai *(stačiakampis, su 2 mažais pailgais stačiakampiai esančiais ant didelio stačiakampio kairės linijos)*
- Sistemos išdėstymo mazgai (deployment nodes)
- Interfeisai
- Priklausomybės
- Paketai
- Pastabos
- Ribojimai

Kokia yra komponentų diagramų paskirtis? (4)
Komponentų diagrama parodo iš kokių fizinių komponentų yra sudaryta sistema.

Komponentas - PĮ vienetas, kuris turi būti išskirstytas:

- pradinio kodo komponentas,
- vykdymo meto komponentas,
- vykdomasis komponentas.

Kad būtų galima parodyti, kokie komponentai kokiame tinklo mazge (kompiuteryje) vykdomi, komponentų diagramose leidžiama kombinuoti sistemos išdėstymo mazgus ir komponentus.

Kas vadinama komponentu? Kaip komponentų diagramose yra modeliuojami komponentai? (4)
Komponentai - tai vykdomi kodo blokai, teikiantys vieną ar daugiau paslaugų, kiekviena iš kurių per paprastą interfeisą pateikia tam tikrą funkcionalumą. Interfeisai išlieka nepakeisti net ir tuomet, kuomet paslaugos teikiamas funkcionalumas pakinta. Komponentų diagramose vaizduojami tik tipai, ne konkretūs komponentų egzemplioriai.

Kuo skiriasi komponentų modeliavimas komponentų ir išdėstymo diagramose. (4)
Išdėstymo diagramos parodo fizinius PĮ ir TĮ komponentų sąryšius, kur yra patalpinti išskirstytos sistemos komponentai. tuo tarpu komponentų diagramose parodomi loginiai ryšiai, kvietimai per interfeisus.

Kas vadinama sistemos išdėstymo mazgu? Kaip komponentų diagramose yra modeliuojami tokie mazgai? (4)
Išdėstymo mazgas vaizduojamas staciakampiu, jame dedami komponentai fiziškai jam priklausantys. pvz. mazgas gali būti : web serveris, jo viduje guli komponentai: IIS servise.

Kuo skiriasi sistemos išdėstymo mazgų modeliavimas komponentų ir išdėstymo diagramose? (4)
Išdėstymo(Netiksliai aprašyta):

- Mazgai vaizduoja tik vykdymo metu egzistuojančius komponentus. Netiksliai aprašyta
- Kokią nors sistemos dalį vykdantis vienetas, paprastai, kompiuteris ar kokia nors kita techninė įranga.
- Mazgų viduje gali būti parodyti komponentų egzemplioriai; taip parodoma, kad komponentas yra patalpintas tame mazge arba jame veikia.

Paaiškinkite kuo skiriasi komponentai ir paketai. (4)
Paketai:

- Naudojami struktūrizuoti UML modelius.
- UML pradinio teksto blokas.

Komponentai:

- "Fizinė, keičiama modeliuojamos sistemos dalis, įgyvendinanti tam tikrą interfeisų rinkinį ir tenkinanti tais interfeisais apibrėžtus reikalavimus",
    - *Pvz.: COM+ komponentai, CORBA komponentai, .NET komponentai, Java Beans...*

Paaiškinkite kuo skiriasi komponentai ir klasės. (4)
Ir klasės, ir komponentai turi pavadinimus ir realizuoja interfeisus.

Klasė:

- Loginė abstrakcija.
- Atributų ir operacijų agregatas.

Komponentas:

- Fizinis daiktas realiai egzistuojantis kompiuteryje.
- Fiziškai pakuoja logines abstrakcijas (klases, interfeisus, …)
- Turi tik operacijas (pasiekiamas per komponento interfeisą).

Išvardinkite išdėstymo diagramos elementus ir paaiškinkite kaip jie vaizduojami. (5)
Sistemos išdėstymo mazgai, jungtys *(connections)*, komponentai, interfeisai, priklausomybės, apibendrinimo ryšiai, paketai, pastabos, ribojimai, asociacijos, agentai

Kokia yra išdėstymo diagramų paskirtis? (4)
Išdėstymo diagrama parodo, kaip kompiuterių tinkluose fiziškai išdėstomi sukurtos sistemos komponentai, t.y. parodo, kokia techninė įranga naudojama sistemai vykdyti ir kaip visa tai yra susieta į vieną visumą

Parodo vykdymo metu veikiančios kompiuterinės sistemos konfigūraciją ir ten veikiančius programinius procesus:

- Diagrama parodo kaip sistema yra išdėstyta tinkluose.
- Ji parodo fizinius PĮ ir TĮ komponentų sąryšius.
- Ji parodo kur yra patalpinti išskirstytos sistemos komponentai.

Kaip komponentai yra modeliuojami išdėstymo diagramose? (4)
Išdėstymo diagramose vaizduojami konkretūs komponentų egzemplioriai.

Kas gali būti išdėstymo mazguose, modeliuojamuose išdėstymo diagramose? (? (4)
Mazguose gali būti ne tik skaičiuojamieji resursai, įskaitant mechaninius, bet ir žmogiškieji. Mazgai gali vaizduoti tiek tipus, tiek ir egzempliorius.

Ką parodo išdėstymo mazgų jungtys? Kaip tokios jungtys yra modeliuojamos UML priemonėmis? (4)
Mazgų jungtys parodo sistemos komunikavimo maršrutus *(interaction paths)*.

Mazgai gali būti jungiami su kitais mazgais asociacijomis. Asociacijoms galima nurodyti rūšis, apibūdinančias komunikavimo maršruto pobūdį.

Išvardinkite paketų diagramos elementus ir paaiškinkite kaip jie vaizduojami.. (3)
Paketai, priklausomybės, pastabos, ribojimai.

nepaaiškinta, kaip naudojami

Kokia yra paketų diagramų paskirtis? (4)
Paketų diagramos skirtos modeliuoti fizinius objektinių sistemų aspektus. Diagramoje parodoma paketų rinkinio organizacija ir priklausomybės tarp paketų.

Kas UML kalboje yra vadinama paketu? Kam naudojami paketai? Kaip jie vaizduojami UML? (3)
Turinti pavadinimą konstrukcija, apjungianti kokį nors UML konstrukcijų (įskaitant kitus paketus) į vieną visumą.

Priklausomybės sąryšiai tarp paketų - atsiranda tuomet, kuomet yra priklausomybė tarp kokių nors tų paketų klasių.

Kaip aprašomas paketo elementų matomumas paketo išorėje? Kokios yra matomumo rūšys? (2)
Paketo elementų matomumas iš paketo išorės aprašomas nurodant prieš elemento vardą matomumo žymenį:

- '+' (public),
- '-' (private),
- '#' (protected).

Kas vadinama paketo prieigos maršrutu? Kokiems tikslams ir kaip yra naudojami prieigos maršrutai? (2)
Prieigos maršrutas *(pathname)* nusako kaip, pradedant nuo sistemos šakninio paketo (arba nuo kurio nors kito taško) prieiti iki norimo modelio elemento

Prieigos maršrutą sudaro paketų vardai, skiriami vienas nuo kito ženklu '::'

Kas UML™ kalboje yra vadinama pastaba? Kokiems tikslams yra naudojamos pastabos? (2)
Specialus grafinis žymuo, kuriame galima pateikti tekstą ir kitokią informaciją.
Tekstas gali aprašyti ribojimus, komentarus, metodo kūną arba žymėtąsias reikšmes.
Pastabos su jose aprašomu modelio elementu (objektu, ryšiu ir kt.) susiejama specialiu pastabos ryšiu.

Kokiems tikslams UML™ yra naudojami ribojimai? Kaip užrašomi ribojimai? Kas tai yra OCL? (4)
Ribojimai aprašo prasminius modelio elementų ryšius arba, kitaip tariant, teiginius bei predikatus, kurie privalo būti teisingi. Jei taip nėra, modeliuojama sistema tampa nekorektiška. UML neapima tokių pažeidimų pasekmių vertinimo.

Kai kurie ribojimai *(pvz. „OR" ribojimas asociacijoms)* yra apibrėžti pačioje UML, kitus turi apibrėžti modeliuotojas.

Ribojimai visuomet užrašomi skliausteliuose ({}).
Ribojimus galima aprašyti laisvos formos tekstu arba specialiai tam skirta OCL kalba.
Su atitinkamu modelio elementu ribojimas susiejamas pastabos ryšiu.

Kas UML™ yra vadinama konstrukcijos rūšimi? Kam UML™ reikalingos konstrukcijų rūšys? (2)
Rūšis *(stereotype)* apibrėžia naują UML konstrukcijų klasę, kuria galima naudotis tame modelyje, kuriame ji apibrėžta.

Galima apibrėžti tik kokios nors jau egzistuojančios konstrukcijų klasės poklasį. Dažniausiai poklasio konstrukcijos nuo klasės konstrukcijų skiriasi jų panaudojimo taisyklėmis. Tokios konstrukcijos gali turėti papildomus ribojimus.

Rūšys *(stereotypes)* yra vienas iš UML plėtros mechanizmų. Rūšis užrašoma tarp prancūziškų kabučių

Kokiems tikslams UML™ yra naudojami savybių sąrašai? (3)
Beveik visoms UML konstrukcijoms kalboje jau yra apibrėžtos tam tikros savybės

Panaudojant žymėtųjų reikšmių mechanizmą galima apibrėžti papildomas savybes

8 paskaita

Kas vadinama užduoties modeliu? (2)
Užduočių modelį sudaro visų su sistema dirbančių agentų ir visų tų agentų vykdomų užduočių visuma. Kad būtų galima aprašyti visos sistemos veikimą, reikia susieti tarpusavyje visus scenarijus, kurie gali įvykti vienos sąveikos metu. Būtent toks scenarijų rinkinys ir modeliuoja užduoties vykdymą. Taigi, užduoties vykdymo modelis yra hierarchinis modelis.

Kokį vaidmenį užduočių modeliuose vaidina pirminis agentas? Koks vaidmuo tenka antriniams agentams? Kas vadinama vidiniu agentu? Kaip visi tie agentai yra susieti tarpusavyje? (4)
Pirminis agentas inicijuoja sąveiką su sistema tam, kad pasiektų tam tikrus savo tikslus. Antrinis agentas – tai toks agentas, be kurio pagalbos sistema negali įvykdyti jos vykdomų užduočių. Vidiniu agentu yra arba nagrinėjamoji sistema, arba koks nors jos posistemis arba kokia nors dar smulkesnė jos dalis. Kad pasiektų savo tikslus, jie vykdo tam tikrus veiksmus. Veiksmas susieja vieno agento tikslą su kito agento atsakomybe.

Kaip užduočių modeliuose suprantama agento atsakomybė? (3)
Įsipareigojimas vykdyti priskirtą užduotį iki jos sėkmingos pabaigos. Kiekvienas agentas turi atsakomybių rinkinį. Kad realizuotų savo atsakomybes, agentai formuluoja atitinkamus tikslus. Kad pasiektų savo tikslus, jie vykdo tam tikrus veiksmus.

Kas vadinama pirminio agento tikslu? (2)
Tai, dėl ko pirminis agentas inicijuoja sąveiką su sistema. Tikslo formuluotė nusako sistemos funkcijas jos panaudojimo terminais, t.y. suprantamai ir taip, kad galima patikrinti, ar jos veikia.

Kas užduočių modeliavime yra vadinama alternatyviaisiais veiksmais? (2)
Kitas būdas pirminio agento tikslui pasiekti, jei agentas, kuriam pateikta užduotis, dėl kokių nors priežasčių nesusidoroja su savo atsakomybe.

Kas užduočių modeliavime yra vadinama sąveika? Kuo skiriasi sąveikų seka ir scenarijus? (3)
Paprasčiausiu atveju sąveika vyksta pasiunčiant pranešimą.

- Pvz., “Spausdinti(reikšmė)"

Tačiau tokių sąveikų seka taip pat yra sąveika. Saveikų seka ir scenarijus nesiskiria. Seka neturi nei jokių išsišakojimų, nei jokių alternatyvių žingsnių. Ji aprašo kas vyko ir kas dar vyks ir tai yra padaroma nurodant atitinkamas sąlygas. Tokios sąveikų sekos yra vadinamos scenarijais.

Paaiškinkite, kas vadinama užduoties vykdymo sritimi. Kuo skiriasi strateginė ir sistemos vykdymo sritys? (3)
Vykdymo sritis(scope) nurodo kokią sistemą nagrinėjame. Strateginiai yra tokie tikslai, kurie yra svarbūs visai organizacijai. Užduotis yra strateginė, jeigu ji parodo kaip pasinaudoti kuriamąja sistema, kad kažką laimėtų visa organizacija. Strateginės vykdymo srities sąvoka vartojama tik verslo modeliavimo lygmenyje. Sistemos vykdymo srities tikslai (užduotys) aprašo funkcionalumą, kurį turi turėti kuriamoji programų sistema. Tokios užduotys dažniausiai yra strateginių užduočių realizavimo žingsniai. Programų sistemos vykdomų užduočių sąvoka yra vartojama atliekant koncepcinį sistemos projektavimą.

Paaiškinkite, kas vadinama užduočių modeliavimo hierarchija. (3)
Viršutiniame modelio lygmenyje yra užduočių diagrama. Kiekviena toje diagramoje pavaizduota užduotis yra dekomponuojama į použdutis (t.y. pirminio agento tikslas yra dekomponuojama į potikslius) ir aprašoma scenarijumi, kuris yra pavaizduojamas sekų diagrama. Scenarijaus žingsniai (sąveikos sekų diagramoje) yra tapatinami su použdutimis. Kai kurios použdutys gali būti paprasti pranešimai.

Použdutys yra žemesnio lygmens užduotys. Todėl kiekvienam scenarijui galima sudaryti savą (žemesnio lygmens) užduočių diagramą. Tos použdutys, kurios nėra paprasti pranešimai, vėl gali būti dekomponuotos ir aprašytos scenarijais. Procesas tęsiasi iki tol, kol viskas nėra išreiškiama paprastais pranešimais.

Kas užduočių modeliavime yra vadinama užduoties lygmeniu? Kuo skiriasi vartotojo tikslas, sumarinis tikslas ir subfunkcija? (3)
Bet kuri iš užduočių turi būti apibrėžta viename iš šių lygmenų: sumarinis tikslas, vartotojo tikslas, subfunkcija.

Vartotojo tikslas atitinka tai, kas yra vadinama “vartotojo užduotis" arba “elementarus verslo procesas“.

Subfunkcijos traktuotinos kaip užduočių modeliavimo paprogramės (t.y. pagalbiniai techninio pobūdžio veiksmai).

Kaip išvengti scenarijaus sprogimo? (3)
Scenarijaus sprogimui išvengti yra naudojamos trys technikos: použdutys (***«include»*** priklausomybė), plėtiniai (***«extend»*** priklausomybė), variantai.

Kaip susieti į vieną visumą visus su viena užduotimi susietus scenarijus? Kuo skiriasi pagrindinis ir alternatyvieji scenarijai? Kas vadinama atkuriamasis scenarijus? (4)
Sėkmės ir nesėkmės scenarijus patogu struktūrizuoti pasinaudojant kelnių su sėkmės ir nesėkmės klešnėmis modeliu. Pagrindinis scenarijus yra paprasčiausias scenarijų, toks, kurį vykdant viskas vyksta gerai ir tikslas pasiekiamas nesusiduriant su jokiomis kliūtimis, t.y. visos žemesnio lygmens použdutys pabaigiamos sėkmingai. Kiti sėkmės scenarijai vadinami atkuriamaisiais (recovery) scenarijais. Scenarijai, kuriuos vykdant tikslas nėra pasiekiami, vadinami nesėkmės scenarijais. Atkuriamieji scenarijai ir nesėkmės scenarijai yra vadinami alternatyviaisiais scenarijais.

Kaip įterpti užduočių modeliavimą į bendrą reikalavimų inžinerijos procesą? Kaip užduočių modeliai siejasi su funkciniais sistemos reikalavimais ir su vartotojo interfeiso reikalavimais? Kaip jie siejasi su sistemos testais? (3)
Užduočių modeliai padeda suprasti kaip vartotojai įsivaizduoja būsimos sistemos funkcinius reikalavimus; yra išeities tašku pradedant nustatinėti, kokios klasės, ryšiai ir būsenos yra reikalingi būsimoje sistemoje; yra pagrindas vartotojo interfeisui projektuoti; yra išeities taškas būsimos sistemos testams projektuoti.
Užduočių modelis nusako visą būsimos sistemos funkcionalumą: sistemos elgseną, kaip ją mato išoriniai stebėtojai.

Paaiškinkite, kuo skiriasi užduočių modeliai verslo ir programų sistemoms. (3)
Kuomet užduočių modeliai yra rengiami modeliuojant verslo procesus, nagrinėjamoji sistema yra tapatinama su visa tą verslą vykdančia organizacija: suinteresuotomis šalimis šiuo atveju yra bendrovės akcininkai, klientai, tiekėjai, o taip pat tos veiklos srities priežiūra užsiimančios valstybinės įstaigos; pirminiais agentais šiuo atveju yra bendrovės klientai (užsakovai) ir galbūt jos tiekėjai. Kuomet užduočių modeliai yra rengiami kuriamai programų sistemai, nagrinėjamoji sistema yra ta programų sistema: suinteresuotomis šalimis šiuo atveju yra žmonės, kurie naudosis programų sistema, bendrovė, kurios nuosavybe bus ta sistema, sistemos probleminę sritį kuruojančios vyriausybinės įstaigos ir kitos programų sistemos; pirminiais agentais šiuo atveju yra tiesioginiai sistemos vartotojai ir galbūt kitos programų sistemos besinaudojančios nagrinėjamąja sistema.

9 paskaita

{} - komentarai

4. Kas ir kokiais tikslais skaito reikalavimus? (2)
{ Visi tikslai išgalvoti, todėl juos reiktų sukonkretizuoti }

Vartotojo reikalavimus skaito: Užsakovo vadovybė (susipažinti su užsakoma sitema), Sistemos vartotojai (susipažinti paviršutiniškai su sistema), Dalykinės srities specialistai (patikrinti ar sistemos reikalavimai neprieštarauja jų darbo pobūdžiui), Sistemos architektas (susipažinti su sistema ir įsivaizduoti kaip ji atrodys). Sistemos reikalavimus skaito: Sistemos vartotojai (patikslintų sistemos reiklalavimus), Dalykinės srities specialistai (patikslintų sistemos reiklalavimus), Sistemos architektas (patikslintų sistemos reiklalavimus). Projektinius reikalavimus skaito: Sistemos eksploatavimo tarnybos (susipažinti su abstrakčiais programos įgyvendinomo būdais), Sistemos architektas (patikslintų sistemos reikalavimus), PS inžinieriai (suprastų ir žinotų kokie išeities ribojimai).

5. Dėl kokių priežasčių verta *formuluoti reikalavimus*? (2)
Pradedant programavimo darbus, neturint nei sistemos, nei projektinių reikalavimų, tikimasi sutaupyti laiko ir pinigų. Nors projekto pradžioje (netgi iki atiduodant sistemą užsakovui) dažnai atrodo, kad tai pavyko pasiekti, iš tiesų tai yra tik iliuzija. Ilgesniame laiko periode toks darbo stilius niekuomet nepasiteisina.

- Viskas, ką pavyksta sutaupyti tokiu būdu, yra prarandama dėl žemos sistemos kokybės ir sunkumų ką nors joje pakeisti.

Gerai suformuluoti reikalavimai taupo pinigus, gerina projekto kokybę.

6. Kokios yra gerai suformuluoto *reikalavimo savybės*? (1)
Gerai suformuluoti reikalavimai turi tenkinti tokias savybes:

- Abstraktus
- Išsamus
- Tikslus
- Vienareikšmis
- Verifikuojamas
- Įgyvendinamas
- Integruojamas
- Lokalizuojamas
- Trasuojamas
- Unikalus
- Glaustas
- Suprantamas

9. Plačiau apibūdinkite *tikslų* reikalavimą? (1)

- Negalima vartoti žodelių „*maždaug*“, „*beveik*“, „*apytiksliai*“ ir pan. Negalima taip pat vartoti griežtai neapibrėžtų terminų „*turi būti patogus*”, „*naudoti nedaug atminties*” , “*veikti greitai*” ir pan.
- Suformuluoti tikslius reikalavimus yra nelengva:
- Formalios specifikacijos yra tikslios, bet jas sudėtinga parašyti ir sunku skaityti.
- Natūraliąja kalba paprasta rašyti ir tokia kalba parašytus reikalavimus lengva skaityti, bet jie dažniausiai yra netikslūs.

10. Patarimai *vienareikšmiui* reikalavimui? (1)
Venk natūraliosios kalbos daugiaprasmiškumo:
Pvz. reikalavimą: “*Kontrolinę sumą reikia imti iš paskutinio įrašo*“ galima interpretuoti 3 skirtingais būdais:

- iš failo gale esančio įrašo;
- iš vėliausiai įrašyto įrašo;
- iš paskutinio apdoroto įrašo.

12. Koks reikalavimas vadinamas *įgyvendinamu*? (1)
Reikalavimas vadinamas įgyvendinamu, jei yra žinomas ir prieinamas toks ekonominiu, juridiniu bei kitais požiūriais priimtinas technologinis procesas, kurio inovaciniai slenksčiai gali būti pašalinti per priimtiną laikotarpį ir už priimtiną kainą ir kurį taikant galima sukurti sistemą, turinčią tuo reikalavimu specifikuojamą savybę.

14. Ar visi reikalavimas gali būti *lokalizuojami*? (1)

- Ne visi reikalavimai yra lokalizuojami. Sistemos aplinkos reikalavimai, projektavimo standartai ir kai kurie kiti reikalavimai galioja visiems sistemos komponentams ir negali būti lokalizuoti nei viename iš jų.

15. Kaip realizuojamas *trasuojamumas*? (1)

- Egzistuoja daug reikalavimų identifikavimo būdų. Vienas iš efektyviausi yra vartoti raidžių ir skaitmenų kombinaciją. Raidinė identifikatoriaus dalis nurodo reikalavimo rūšį (tam pakanka fiksuoto raidžių skaičiaus), o skaitmeninė - tos rūšies reikalavimo numerį.
- Trasavimas reikalingas reikalavimų valdymui. Padarius kokius nors reikalavimų pakeitimus, trasuojamumas įgalina nustatyti, kokie aukštesniųjų ir žemesniųjų lygmenų reikalavimai yra susiję su keičiamu reikalavimu ir patikrinti, ar jų taip pat nereikia keisti.

19. Ką nusako *funkcinių* ir *nefunkcinių* reikalavimai, jų skirtumai. (2)

Funkciniai reikalavimai nusako, kokias paslaugas privalo teikti sistema, koks turi būti jos reakcijos laikas i konkrečius sistemos veiksmus ir kaip ji privalo elgtis konkrečiose situacijose. Nefunkciniai reikalavimai ??? tai ribojimai kurie riboja sistemos darbą. Pavyzdžiu: nefunkciniai reikalavimai apima ribojimus sistemos paslaugų teikimo būdui(maksimali trukmė), taip pat su šiais reikalavimai įgyvendinamos funkcijos(tokios kaip patikimumas), bei standartai kurių privalu laikytis. Projekto vykdymo reikalavimai priskiriami nefunkciniams reikalavimams.

20. Paaiškinkite sistemos *pagrindinio funkcionalumo* ir jos *pagalbinio funkcionalumo* skirtumus. (2)

Funkciniai reikalavimai aprašo:

- pagrindinį (probleminį) sistemos funkcionalumą;
  - t.y. funkcionalumą, kurio reikia patenkinti vartotojų operacinius poreikius, nusakytus užduotimis koncepciniame verslo modelyje.
- pagalbinį funkcionalumą;
  - t.y. funkcionalumą, kurio reikia sistemai aptarnauti, prižiūrėti, administruoti arba darbui su ja palengvinti.

21. Kaip nustatyti, kokius funkcinius reikalavimus turi tenkinti sistema? (3)

- Funkciniai reikalavimai turi aprašyti visas sistemos įgyvendinamas funkcijas.
  - Viena vertus, funkcijos turi tenkinti poreikių specifikacijoje aprašytus operacinius vartotojų poreikius.
  - Kita vertus, funkciniai reikalavimai yra išvedami iš verslo koncepciniame modelyje specifikuotų užduočių (tiksliau, tų užduočių, kurias nuspręsta kompiuterizuoti).
    - Koncepciniame verslo modelyje specifikuotų užduočių aibė nėra išsami, nes ji neapima sistemos administravimo, aptarnavimo bei priežiūros užduočių.
    - Tos užduotys turėtų būti išvedamos iš sistemos naudojimo scenarijaus.

22. Kokią informaciją reikia pateikti, aprašant funkcinį reikalavimą? (2)

- funkcijos įeigą (pradinius duomenis);
- funkcijos išeigą (rezultatą);
- principinį algoritmą (jei to reikia).
  - Pradiniai duomenys ir rezultatai turi būti aprašyti detalumu, pakankamu parašyti programą.
  - Algoritmas reikalingas tam, kad būtų galima suprasti, ką funkcija daro. Tačiau tai nereiškia, kad funkciją reikia realizuoti naudojant būtent tokį algoritmą.

23. Kokia programų sistema vadinama *korektiška*? Kada ji praranda savo korektiškumą? (1)

Programų sistema vadinama korektiška, jei ji tenkina operacinius vartotojų poreikius ir funkcinės savybės atitinka reikalavimų specifikacija numatytus reikalavimus.

1. Jei programų sistema netenkina jos funkcinių reikalavimų, ji daro ne tai, ko iš jos tikimasi ir tampa nekorektiška.

24. Kokias ribojimų grupes apima nefunkciniai reikalavimai? Apibūdinkite jas trumpai. (3)

{Visi ribojimai pateikti plačiau sekančiuose klausimuose}

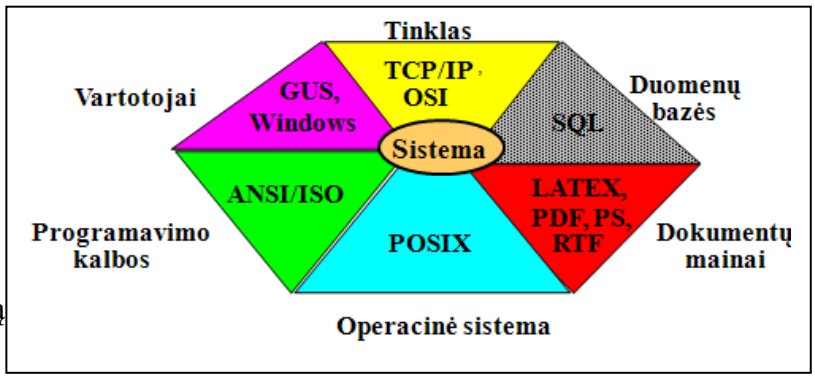
Nefunkciniai reikalavimai aprašo:

- interfeiso ribojimus;
- veikimo ribojimus;
- ekonominius ribojimus;
- politinius ribojimus;
- teisinius ribojimus.

25. Kokius programų sistemos aspektus riboja *interfeiso ribojimai*? Kas įvyksta juos pažeidus? (2)

Interfeiso ribojimai riboja sistemos ir jos aplinkos sąveikos būdus.

- Programų sistemos aplinką sudaro:
  - vartotojai;
  - kitos dalykinės programų sistemos;
  - operacinė sistema;
  - duomenų bazės;
  - kompiuterių tinklas;
  - programavimo kalbos;
  - dokumentai.



- Interfeiso reikalavimai nieko nekalba apie tai, ką
- sistema daro. Jie aprašo, kaip tomisgalimybėmis
- galima pasinaudoti.
- Pažeidus interfeiso reikalavimus, sistema išlieka korektiška, bet ji praranda sąveikos su tam tikrais savo aplinkos elementais (vartotojais, OS, DB ir t.t.) galimybes.
  - Nors sistema ir vykdo visas numatytas funkcijas, tačiau jomis nebegalima pasinaudoti.

26. Kokias reikalavimų grupes apima interfeiso reikalavimai? (1)

Interfeiso ribojimai:

- vartotojų interfeisų reikalavimai;
- sąveikos su operacine sistema reikalavimai;
- sąveikos su duomenų bazėmis reikalavimai;
- sąveikos su kompiuterių tinklais reikalavimai;
- sąveikos su programavimo aplinkomis reikalavimai;
- dokumentų mainų reikalavimai;
- sąveikos su kitomis dalykinėmis programų sistemomis reikalavimai.

27. Kokias reikalavimų grupes apima vartotojo interfeiso reikalavimai? (1)

- Vartotojo interfeiso reikalavimai:
  - užduočių formulavimo kalbos (UFK) reikalavimai;
  - naudojimo paprastumo (easy-to-use) reikalavimai;
  - patogumo vartotojui (user-friendless) reikalavimai;
  - ergonominiai reikalavimai.

28. Kas tai yra užduočių formulavimo kalbos *semantinė galia*? Kokiais reikalavimais ji nusakoma? (2)

UFK*(vartotojo interfeisas)* semantinė galia:

- UFK konceptų abstrakcijos lygmuo
  - Nelabai kvalifikuoto specialisto požiūris: "*Tai per daug bendra, kad būtų galima suprasti, ką gi konkrečiai padarys sistema pateikus tokią užduotį.*"
  - Kvalifikuoto specialisto požiūris: "*Tai per daug konkretu. Formuluodamas tokias užduotis, aš susipainioju detalėse.*"
  - Dauguma žmonių masto tam tikro abstrakcijos lygmens terminais ir nervinasi, jei jie yra verčiami mąstyti kito lygmens terminais.
- Viskas priklauso nuo konkretaus vartotojo: viskas turi būti kuo paprasčiau, bet ne per daug supaprastinta!

29. Nuo ko priklauso užduočių formulavimo kalbos *procedūriškumo* laipsnis? Kokiais reikalavimais jis nusakoma? (4)

- Kalbos procedūriškumo laipsnį lemia naudojamas skaičiavimų modelis. Pagal procedūriškumo laipsnį skiriamos deklaratyviosios, procedūrinės ir mišriosios kalbos.
- Procedūrinėmis kalbomis užduotys formuluojamos kaip komandų sekos. Komandų vykdymo tvarka nusakoma valdymo struktūromis. Informaciniams komandų ryšiams nurodyti naudojama specialiai tam tikslui skirtos struktūros, vadinamos kintamaisiais. Kintamųjų visuma sudaro vadinamąją dinaminę atmintį. Įvykdžius komandą, kai kurių kintamųjų reikšmės pakinta, t.y. pakinta dinaminės atminties būsena. Formuluojant užduotį procedūrine kalba, vienaip ar kitaip nurodomas ir tos užduoties realizavimo būdas.
- Deklaratyviosios kalbos neturi nei kintamųjų, nei dinaminės atminties. Jose naudojami tik parametrai, pagal kurių reikšmes nusakomi pradiniai vykdomos užduoties duomenys. Vykdant užduotį, parametrų

reikšmės nekinta. Deklaratyviosiose kalbose nėra ir valdymo struktūrų. Komandų vykdymo tvarka tokiose kalbose yra nesvarbi. Formuluojant užduotis deklaratyviąja kalba, nurodomas tik užduoties vykdymo tikslas. Realizavimo būdas šiuo atveju nespecifikuojamas.

- Mišriosios kalbos užima tarpinę padėtį. Dažniausiai jos yra grindžiamos matematinių atvaizdžių teorija. Užduotys tokiomis kalbomis yra konstruojamos kaip pasirinktų primityvių atvaizdžių kompozicijos. Atvaizdžių kompozicija nusako atvaizdžių taikymo tvarką. Tai - procedūriškumo šaltinis. Galimi ir kiti užslėpti procedūriškumo šaltiniai, pavyzdžiui, kontekstinės priklausomybės. Dažniausiai jos pasireiškia tuo, kad kai kurias leidžiama naudoti tik tam tikrame kontekste, t.y. esant tik tam tikrai vidinei sistemos būsenai.
- Kokį skaičiavimų modelį tikslingiau yra naudoti konkrečioje programų sistemoje, priklauso nuo daugelio faktorių. Svarbiausiais iš jų yra sistemos vykdomų užduočių pobūdis ir naudotojų darbo tradicijos.

30. Kas tai yra užduočių formulavimo kalbos *dalykinė orientacija*? Kas vadinama *metafora*? (4)

Dalykinė orientacija:

- kalbos dalykinė orientacija priklauso nuo metaforos, kuria grindžiama kalba
  - žmonės naujus dalykus perpranta naudodamiesi savo turimomis žiniomis ir patirtimi;
  - bent jau pradžioje, naujas sąvokas stengiamasi išreikšti per jau žinomas sąvokas.

Kalbos dalykinė orientacija:

- vartotojai, bent jau pradžioje, taip pat bando programų sistemą suvokti per tai, ką jie jau žino apie dalykinę sritį, t.y. jie bando mąstyti dalykinės srities metaforos terminais.
- Metaforizavimo principas: Naujas pažinimo struktūras žmonės kuria metaforizuodami esamas (jau išstudijuotas) pažinimo struktūras.
- Kad metafora būtų naudinga, ji turi būti išreikštiniu būdu susieta su tuo, ką norima metaforizuoti t.y. su problemine sritimi.

- Svarbi tokios sąsajos (tiksliau, atvaizdžio) savybė yra jos kongruentiškumas: probleminės srities esybės ir ryšiai turi būti tiesiogiai atvaizduojami į interfeiso esybes (piktogramas, komandų pavadinimus, objektų pavadinimus ir t.t.) ir ryšius taro jų.

{ Jei visa programa ir atskiri jos elementas yra lengvai susiejami su atitinkamais vartotojo aplinkos procesais bei objektais, tai vartotojas pasijunta „kaip žuvis vandenyje“ ir gali laisvai “plaukioti” po sistemą. Visų pirma tai liečia vartotojo interfeisą ir su juo siejamą užduočių formulavimo kalbą. }

31. Kas tai yra UFK selektyvinė geba? Kokiais reikalavimais ji nusakoma? (2)

Kalbos selektyvinė geba:

- Duomenų filtravimas
- Filtrai aprašomi naudojant kvalifikavimo išraiškas

Kalbos selektyvinė geba:

- Kalbos selektyvinė geba priklauso nuo to, kokios kvalifikavimo išraiškos yra leidžiamos užduočių formulavimo kalboje.
  - Selektyvinė geba nusako kokiu tikslumu galima filtruoti įvesties ir išvesties duomenis. Kitaip tariant, ji lemia kokiu tikslumu vartotojas gali aprašyti užduoties pradinius duomenis (imamus iš įvesties srauto arba iš DB) ir užduoties rezultatus *(pvz., ką ir kokia tvarka spausdinti norimoje gauti ataskaitoje)*.

32. Kas tai yra UFK *raiškos geba* (*išsamumas)*? (2)

Kalbos raiškos galia (išsamumas):

- UFK yra išsami jei jos raiškos priemonėmis galima specifikuoti visus užduoties aspektus.
  - Kartais UFK turi priemones tik skaičiuojamiesiems užduoties aspektams aprašyti, nes kiti aspektai (pvz., duomenų rašymas į bazes ar ataskaitų generavimas) yra aprašomi kitomis kalbomis.
  - Kartais, kuriant UFK, nepavyksta tiksliai specifikuoti visų galimų užduočių aibę. Taigi, neįmanoma ir tiksliai numatyti, kokių priemonių prireiks užduotims specifikuoti. Tokiais atvejais kalboje numatoma išplėsties taško konstrukcija ir leidžiama prie užduoties aprašymo jungti plėtinius parašytus skriptų programavimo kalbose (Perl, JavaScript ir kt.)

33. Ką apibūdina UFK *sintaksiniai reikalavimai*? (2)

UFK*(Vartotojo interfeisas)* sintaksė. UFK gali būti suprojektuota kaip:

- tekstinė kalba (pvz., sakiniai, primenantys lietuvių kalbos sakinius);
- meniu kalba;
- formų kalba;
- grafinė kalba;
- piktogramų kalba;
- hiperteksto kalba
- ir t.t.

34. Kas tai yra UPP? (2)

UPP:

- Užduoties aprašas yra pateikiamas sistemai pagal taisykles, numatytas atitinkamu protokolu. Formuluojant interfeiso ribojimus turi būti suformuluoti ir užduočių pateikties protokolo reikalavimai.
- UPP aprašo, kokiais pranešimais, pateikdamas užduotį, vartotojas keičiasi su sistema, kokia eilės tvarka pranešimai yra pateikiami ir kaip, pasinaudojant atitinkama technine įranga (klaviatūra, pele ar pan.), pranešimai yra formuluojami ir perduodami.

35. Apibūdinkite programų sistemos *naudojimo paprastumo* reikalavimus. (3)

Naudojimo paprastumo (easy-to-use) reikalavimai:

- ***Vidinė darna:*** komandų formato standartas, klavišų naudojimo nepriklausomybė nuo konteksto, pranešimuose vartojamų terminų darna, manipuliavimo ekranu operacijų standartas ir t.t..
- ***Išorinė darna:*** atitikimas išoriniams standartams (pvz., GUI standartams).
- ***Akivaizdumas:*** interfeisas suprojektuotas metaforos terminais.
- ***Informatyvumas:*** informuoja apie padarytų klaidų pobūdį, aiškina kaip elgtis ir pan.
- ***Prasmingumas:*** visi sistemos pranešimai yra glausti, informatyvūs ir formuluojami metaforos terminais.

36. Apibūdinkite programų sistemos *patogumo vartotojui* reikalavimus. (2)

Patogumo vartotojui (user-friendliness) reikalavimai:

- ***Informavimo priemonių tinkamumas** (helpfulness)***:** ką reikia įtraukti į informavimo (help) failus ir kaip tie failai turi būti organizuoti.
- ***Patrauklumas:*** spalvos ir kiti dizaino ypatumą.
- ***Individualizavimas:*** kaip vartotojas gali individualizuoti (prisitaikyti savo pomėgiams) interfeisą.

37. Apibūdinkite programų sistemos *ergonominius* reikalavimus. (1)

Ergonominiai reikalavimai:

- kaip atsižvelgti į potencialių vartotojų psichofiziologines charakteristikas *(pvz., aklus vartotojus)*.

38. Kokius programų sistemos aspektus riboja *veikimo ribojimai*? Kas atsitinka juos pažeidus? (2)

Veikimo ribojimai apima (taigi ir riboja aspektus):

- tikslumo reikalavimus;
- patikimumo reikalavimus;
- gyvybingumo reikalavimus;
- robastiškumo reikalavimus;
- našumo reikalavimus.

Pažeidus veikimo reikalavimus, sistema išlieka korektiška. Nėra pažeidžiama ir jos sąveika su aplinka. Tačiau yra prarandama jos operacinė kokybė *(pvz., ji veikia lėčiau, yra mažiau patikima ar labiau pažeidžiama)*.

39. Kokias reikalavimų grupes apima *veikimo ribojimai*? (1)

Veikimo ribojimai apima:

- tikslumo reikalavimus;
- patikimumo reikalavimus;
- gyvybingumo reikalavimus;
- robastiškumo reikalavimus;
- našumo reikalavimus.

40. Apibūdinkite, kas tai yra programų sistemos *tikslumo* reikalavimai. (2)

Tikslumas:

- Nepakanka vien tik pasakyti, kad programų sistema turi mokėti paskaičiuoti, tarkime, įmonės mėnesines išlaidas gamybai, reikia apibrėžti kokiu tikslumu reikia paskaičiuoti, ką sistema daro kuomet trūksta duomenų, bei kokie turi būti kokybiniai tos funkcijos parametrai.
- Duomenų vaizdavimo tikslumas:
  - Programų sistemos geba skirti panašius objektus vieną nuo kito.
- Skaičiavimų tikslumas:
  - Nusako, kokio dydžio skaičiavimų paklaidos yra leistinos sistemoje.
  - Kitaip tariant, nusako rezultatų tikslumą.
- Į sistemą vedami duomenys turi tenkinti verslo taisykles.
  - Verslo taisyklės reglamentuoja duomenų tikslumą ir išsamumą.
  - Tai reiškia, kad, kuriant sistemą, reikia išreikštiniu būdu suformuluoti verslo taisykles ir jas teisingai taikyti.
- Kokio tikslumo reikia, priklauso nuo sistemos paskirties ir jos realizuojamų funkcijų pobūdžio.
  - Jei didelio tikslumo nereikia, nėra jokios prasmės reikalauti, kad, tarkime, skaičiavimo paklaidos neviršytų 0.00001.
  - Pasiekti tikslumą kainuoja, todėl reikia reikalauti tokio tikslumo, kokio iš tiesų reikia, bet ne didesnio.

41. Apibūdinkite, kas tai yra programų sistemos *patikimumo* reikalavimai. (2)

Patikimumas:

- Programų sistemos trykių neigiamo poveikio vartotojų verslo tikslams dydis.
- Tai normalaus veikimo trykių dažnio ir rimtumo matas.
- Matų pavyzdžiai:
  - vidutinis laikas tarp dviejų trykių;
  - tikimybė, kad sistema nebus galima pasinaudoti;
  - vidutinis trykių dažnis;
  - dėl trykių prarandamo laiko procentas.

42. Apibūdinkite, kas tai yra programų sistemos *gyvybingumo* reikalavimai. (1)

Programų sistemos geba apsaugoti jos kritines funkcijas nuo trykių poveikio.

- Kokiu mastu sistema gebės vykdyti kritines (gyvybines) funkcijas trykiams pažeidus kitas jos dalis.

43. Apibūdinkite, kas tai yra programų sistemos *robastiškumo* reikalavimai. (1)

Programų sistemos geba automatiškai atkurti savo prarastą funkcionalumą, ypač, praradus jį dėl klaidingų duomenų arba dėl kokių nors ypatingų situacijų:

- Nusako, kokiu mastu sistema geba išvengti trykių pateikus jai neteisingus duomenis arba ja neteisingai naudojantis.
  - Tai sistemos “nenumušamumo” matas.

- Matų pavyzdžiai:
  - laikas, reikalingas sistemos funkcionalumui atkurti;
  - trykius iššaukiančių įvykių procentas;
  - tikimybė, kad trykio metu bus sugadinti duomenys.

44. Apibūdinkite, kas tai yra programų sistemos *našumo* reikalavimai. (4)

Našumas - nusako kaip greitai sistema reaguoja į stimulus ir kiek resursų ji sunaudoja tai darydama.

Yra keli našumo aspektai:

- reakcijos laikas *(response time)*;
- pralaidumas *(throughput);*
- produktyvumas *(efficiency)*;
- gaišties laikas *(latency);*
- masto keitimas *(scalability)*.

Reakcijos laikas:

- Reakcijos laiko į įvykius *(behavioural response time)* reikalavimai apibūdina vartotojo sąveiką su sistema

        - Apskritai, vartotojas tikisi, kad sistema atsakys jam per kelias sekundes, nepriklausomai nuo to, ką kompiuteris tuo metu daro.
        - Atsakymas ne būtinai turi būti laukiamas rezultatas, bet sistema turėtų bent jau parodyti, kad ji priėmė vartotojo pateiktą užduotį.
        - Labai svarbu, kad sistema iš karto vienaip ar kitaip atsakytų į bet kokius vartotojo veiksmus.

Reakcijos laikas:

- Rezultatų pateikties laiko (operational response time) reikalavimai
    - Jei programų sistema kompiuterizuoja kokias nors rankines procedūras, kurioms atlikti reikėjo visos savaitės, o dabar pakanka vienos valandos, tai be abejo jau yra didelis pasiekimas.
    - Formuluojant rezultatų pateikties laiko reikalavimus, visų pirma reikia galvoti apie tai, kiek laiko galima skirti atitinkamai verslo operacijai atlikti.
        - Kaip greitai turi veikti sistema, kad būtų pasiekti organizacijos verslo tikslai?

- Gaišties laikas *(latency)*:
    - Kiek laiko prie nurodyto apkrovos (pvz., apdorojamų duomenų apimčių) dydžio gali sugaišti sistema vienam įvykiui apdoroti?
- Pralaidumas *(throughput)*:
    - Kiek įvykių per nurodytą laiko tarpą prie nurodyto apkrovos (pvz., apdorojamų duomenų apimčių) dydžio turi gebėti apdoroti sistema?
- Produktyvumas:
    - Kiek resursų galima sunaudoti nurodytiems pralaidumu ir produktyvumui užtikrinti?
- Masto keitimas *(scalability)*:
    - Kiek papildomų resursų galima sunaudoti nurodytiems pralaidumu ir produktyvumui užtikrinti didinant sistemos apkrovas?

45. Kokius programų sistemos aspektus riboja *ekonominiai ribojimai*? Kas atsitinka juos pažeidus? (2)

- Ekonominiai ribojimai formuluojami siekiant sumažinti ilgalaikes išlaidas sistemai.
    - Pažeidus ekonominius ribojimus, užsakovas ir(arba) vykdytojai be reikalo praranda tam tikras pinigų sumas ir tai atsitinka todėl, kad tam tikros sistemos savybės buvo nepakankamai gerai apgalvotos.
        - Sistema išlieka korektiška, jos sąveika su aplinka nėra pažeidžiama ir ji nepraranda savo veikimo kokybės.
        - Taigi, paprastai sistemos vartotojas tiesiogiai nepajunta jokių ekonominių ribojimų pažeidimo pasekmių.
- Lėšos gali būti prarastos dėl to, kad su sistema sunku dirbti ją diegiant, aptarnaujant ar prižiūrint, arba dėl to, kad jos komponentų negalima tiražuoti t.y. pakartotinai panaudoti kituose projektuose.

46. Kokios reikalavimų grupės išplaukia iš ekonominių ribojimų? (1)

Iš ekonominių ribojimų išplaukia:

- diegiamumo reikalavimai;
- aptarnaujamumo reikalavimai;
- prižiūrimumo reikalavimai;
- tiražuojamumo reikalavimai.

47. Apibūdinkite, kas tai yra programų sistemos *diegimo* reikalavimai. (4)

Diegiamumo reikalavimai:

- ***Instaliuojamumas:***
    - ruošinio patikimumas, instaliavimo trukmė, instaliavimo procedūros, parametrizavimas, konfigūravimo galimybės, resursų poreikis (instaliavimui).
- ***Įsisavinamumas:***
    - Matas: tikimybė, kad per laiką [t1,t2] atitinkamą išsilavinimą turintis asmuo sugebės išmokti dirbti su sistema ar ją administruoti.
- ***Išmokstamumas:***
    - Išmokstamumas priklauso nuo:
        - suprantamumo;
        - sudėtingumo.

- Suprantamumas, savo ruožtu, priklauso nuo:
  - koncepcinės skaidros,
  - virtualumo.
- Programų sistema yra koncepciškai skaidri, jei ji:
  - turi darnų, akivaizdų, informatyvų ir prasmingą vartotojo interfeisą;
  - yra komunikatyvi,
    - t.y. tiek jos įvesties, tiek ir išvesties duomenys yra glausti, prasmingi ir pateikiami vartotojui lengvai suprantama forma.
- Programų sistema vadinama virtualia, jei ji nuo vartotojo slepia kompiuterinę platformą.
  - Tai reiškia, kad sistema galima pasinaudoti neturint jokių (arba tik minimalias) žinių apie techninės įrangos, sisteminės PĮ ar kompiuterių tinklo ypatumus.
- Programų sistemos sudėtingumas priklauso nuo jos realizuojamų funkcijų skaičiaus, tų funkcijų semantinės galios ir jų tarpusavio sąryšių.

- ***Pastangos duomenų bazėms sukurti:***
  - siekiant palengvinti pradinį DB kūrimą, sistemoje reikia numatyti specialiai tam skirtas funkcijas (duomenims skaitmenizuoti, naujoms DB kurti, liktinėms DB transformuoti į naują formatą ir t.t.).

48. Apibūdinkite, kas tai yra programų sistemos *aptarnaujamumo* reikalavimai. (2)

- Aptarnaujamumo reikalavimai:
  - Vartotojų (įskaitant sistemą administruojantį personalą) pastangos, kurių prireikia naudojant sistemą savo užduotims vykdyti.
    - galimybė pritaikyti sistemą prie vartotojo įgūdžių lygmens, sistemos veiksmų siejimas su užduočių vykdymo kontekstu, specialus dažnai vykdomų operacijų palaikymas, "karštieji" klavišai ir t.t.

49. Apibūdinkite, kas tai yra programų sistemos *prižiūrimumo* reikalavimai. (3)

- Prižiūrimumo reikalavimai:
  - Nuo jų priklauso sistemos darnos palaikymo ir jos perdarymų išlaidos.
  - Prižiūrimumas priklauso nuo:
    - ***Taisomumo:*** Kiek pastangų reikia klaidoms pašalinti?
    - ***Keičiamumo:*** Kiek pastangų reikia sistemai pritaikyti prie jos reikalavimų pokyčių?
    - ***Plečiamumo:*** Kiek pastangų reikia sistemos komponentams pakeisti ir jos funkcionalumui išplėsti?
    - ***Perkeliamumo:*** Kiek pastangų reikia sistemai į kitą platformą perkelti?
    - ***Komponuojamumo:*** Kiek pastangų reikia sistemos sąveikai su kitomis sistemomis organizuoti?
    - ***Testuojamumo:*** Kiek pastangų reikia testams sistemos savybėms patikrinti suprojektuoti ir sukurti?

50. Apibūdinkite, kas tai yra programų sistemos *tiražuojamumo* reikalavimai. (1)

Tiražuojamumo reikalavimai:

- kurie komponentai turi būti tiražuojami?
- kokius reikalavimus turi tenkinti tiražuojami komponentai?
  - t.y. kokie komponentų tiražavimo metodai planuojami naudoti.

51. Kokius programų sistemos aspektus riboja *politiniai* ribojimai? Kas atsitinka juos pažeidus? (2)

Politiniai ribojimai:

- Politiniai ribojimai išplaukia iš organizacijos verslo politikos ypatumų. Tai, visų pirma, verslo paslapčių apsaugos klausimai. Sistemai pažeidus verslo ribojimus, yra pažeidžiama organizacijos verslo politika.

52. Apibūdinkite, kas tai yra programų sistemos *apsaugos* reikalavimai? (3)

Apsaugos (security) reikalavimai:

- Apibrėžia, kokiu mastu sistema turi būti apsaugota nuo galimybių ja pasinaudoti neteisėtai.
  - Teisėtu PS naudojimu vadinamas sistemos naudojimas pagal jos paskirtį, atliekamas tam oficialius įgaliojimus turinčio asmens ar proceso (įskaitant kitas programų sistemas).

Apsaugos reikalavimai:

- grėsmės, nuo kurių turi būti apsaugota sistema;

- vartotojų ir procesų registravimo procedūros;
- vartotojų ir procesų skirstymo į klases ir teisių priskyrimo toms klasėms taisyklės;
- vartotojų ir procesų autorizavimo taisyklės;
- duomenų ir sistemos funkcijų klasifikavimo pagal slaptumo kategorijas taisyklės ir tų klasių apsaugos lygmenys.

53. Kokius programų sistemos aspektus riboja *teisiniai* ribojimai? (1)

Teisiniai ribojimai:

- Teisiniai ribojimai reglamentuoja kas leidžiama ir kas neleidžiama daryti kuriamoje PS, atsižvelgiant į galiojančius teisės aktus (asmens duomenų apsauga, statistinių duomenų apdorojimas ir pateiktis, garbinga konkurencija, teisinis konkrečios dalykinės srities reguliavimas ir kt.).
  - Iš teisinių ribojimų išplaukiantys reikalavimai specifikuoja tas PS savybes, kurios garantuoja, kad naudojantis sistema nebus pažeisti kokie nors įstatymai, standartai, vyriausybės nutarimai ir pan.
    - Formuluojami tik tokie reikalavimai, kurie vienaip ar kitaip suvaržo sistemą kuriančių projektuotojų ar programuotojų veikimo laisvę.

54. Ką reglamentuoja ISO 9126 standartas? Kaip jis susijęs su reikalavimų formulavimu ir jų įgyvendinimo vertinimu?(4)

Pagrindinis ISO 9126 standarto tikslas yra apibrėžti gerai žinomus žmonių daromus nukrypimus, kurie gali nepalankiai įtakoti PĮ kūrimo projekto teikimą ir supratimą*, pvz. prioritetų pakeitimas jau prasidėjus projektui ar neturint aiškaus "sėkmės" apibrėžimo*. Išsiaiškinus ir sutarus projekto prioritetus ir vėliau abstrakčius prioritetus *(susitarimą)* pavertus į išmatuojamas vertes *(įvedimo duomenys gali būti patikrinti su X schemoje nulio intervencija)*. ISO 9126 bando išdėstyti, išaiškinti bendrą projekto objektų ir tikslų supratimą.

Quint yra labai svarbus formuluojant PS reikalavimus.

- **Svarba:** Quint pateikia sąvokų sistemą, vartojant kurią PS užsakovai, vartotojai ir tą sistemą kuriantys PS inžinieriai gali diskutuoti apie PS kokybę ir susitarti apie konkrečių kokybės atributų prioritetus.
- Quint ir užduočių modeliavimo panaudojimas specifikuojant nefunkcinius PS reikalavimus:
  - konkrečiai PS ne visi 32 kokybės atributai yra vienodai svarbūs: nustatyk jų prioritetus!
  - formuluok reikalavimus taip, kad būtų galima pamatuoti jų įgyvendinimo laipsnį
    - ne: “Sistema turi būti patogi vartotojui”
    - bet: “Nei vienai užduočiai formuluoti vartotojas neturi gaišti daugiau kaip 2 minutes”.
  - Siek reikalavimus su konkrečiomis užduotimis
    - Pavyzdys: “Programa turi patikrinti PIN kodą per 1 sekundę”.

55. Kokie reikalavimai vadinami *anotuotais*? (2)

- Anotuoti reikalavimai:
  - Reikalavimai anotuojami, norint pateikti sistemą kuriančiai organizacijai tam tikras rekomendacijas.
  - Reikalavimo statusas (E, D, O):
    - **E** - Esminiais vadinami tie reikalavimai, kurių neįgyvendinus negali būti patenkinti operaciniai vartotojų poreikiai.
      - Neįgyvendinus tokių reikalavimų, užsakovui sistema tampa bevertė ir jis tokios sistemos nepriims.
    - **D** - Pageidavimais vadinami reikalavimai, kuriuos įgyvendinus vartotojo darbas palengvėja ar supaprastėja, tačiau be kurių iš principo galima ir apseiti.
      - Užsakovas gali priimti sistemą ir neįgyvendinus kai kurių pageidavimų. Ji bus tinkama tik galbūt mažiau patogi.
    - **O** - Papildomais vadinami reikalavimai, reikalingi papildomiems vartotojų operaciniams reikalavimams tenkinti.
      - Paprastai tokie reikalavimai yra įgyvendinami už papildomą mokestį.
    - Reikalavimo statusas:
      - Vykdytojai gali sugaišti labai daug laiko kokiam nors reikalavimui įgyvendinti ir tik po to sužinoti, kad vartotojui daug svarbiau laiku gauti sistemą netenkinančią šio reikalavimo, negu dviem mėnesiais vėliau gauti sistemą, kuri tą reikalavimą jau tenkina.
      - Nurodžius reikalavimų statusus, tokių situacijų galima išvengti.

- Reikalavimų galiojimo laikas (S,U,T):
  - **S** - niekad nekeičiamas reikalavimas;
  - **U** - toks reikalavimas, kurio keitimo tikėtinumas nelygus nuliui;
  - **T** - tik laikinai *(pvz., kol galutinai bus pereita nuo anksčiau naudotos sistemos prie naujos sistemos)* reikalingas reikalavimas.
- Reikalavimo galiojimo laikas padeda įvertinti, kokios tą reikalavimą realizuojančio kodo savybės (galimybė lengvai keisti, našumas ar kt.) yra svarbesnės.

- ***Kritiškumo laipsnis:***
  - **S** - sunkios pasekmės;
  - **A** - apysunkės pasekmės;
  - **L** - lengvos pasekmės.
- Reikalavimo kritiškumo laipsnį nusako jo pažeidimo pasekmės. Jos gali būti sunkios, apysunkės arba lengvos. Pasekmių sunkumo laipsnis nustatomas pagal materialinės arba kitokios galimos žalos dydį.
  - Pavyzdžiui, sistemos trykis yra kritinis lėktuvą valdančiai sistemai ir mažai kritinis darbo užmokesčio skaičiavimo sistemai. Kita vertus, rezultatų tikslumas yra kritinis darbo užmokesčio skaičiavimo sistemai ir mažiau kritinės lėktuvą valdančiai sistemai.

56. Kokios yra gerai parašytos reikalavimų specifikacijos savybės? Apibūdinkite jas trumpai. (5)

1. Reikalavimų specifikacijos dokumentas gali apimti ir vartotojo reikalavimus (t.y. operacinius poreikius), ir sistemos reikalavimus.
2. Šis dokumentas NĖRA projektinis dokumentas. Jame aprašoma KĄ sistema turi daryti, bet ne tai, KAIP tai turi būti realizuota.

- Gerai parašytos reikalavimų specifikacijos savybės:
  - Visų pirma reikalavimų specifikacija yra gerai parašyta, jeigu ji tenkina projektuotojų poreikius.
    - Tai reiškia, kad dokumento savybes lemia tikslai, kuriems jis yra naudojamas.
    - Vienų savybių reikia tam, kad dokumentą būtų patogu skaityti, kitų tam, kad būtų nesunku patikrinti, ar projektuotojai tikrai įgyvendino visus reikalavimus, dar kitų tam, kad dokumentą būtų nesunku keisti.
  - Nėra kokio nors visuotinai pripažinto reikalavimų specifikacijos formato standarto, tačiau yra įprasta numeruoti visus reikalavimus:

{ 3.5.1 Adding nodes to a design 3.5.1.1 The editor shall provide a facility for users to add nodes of a specified type to their design. 3.5.1.2 The sequence of actions to add a node should be as follows: 1. The user should select the type of node to be added. 2. The user should move the cursor to the approximate node position in the diagram and indicate that the node symbol should be added at that point. 3. The user should then drag the node symbol to its final position. Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning. Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1 }

- **Konceptualumas** *(appropriateness):* reikalavimų specifikacija yra konceptuali, jei visi joje pateikti reikalavimai yra abstraktūs, t.y. joje nėra liečiami sistemos projektavimo ar įgyvendinimo klausimai.
  - Nekonceptuali specifikacija per daug suvaržytų projektuotojus ir programuotojus, trukdytų jiems parinkti geriausią sistemos įgyvendinimo būdą.
  - Kita vertus, dažnai yra labai sunku išvengti kai kurių projektavimo ar realizavimo ribojimų.
- ***Koncepcinė skaidra:*** apima specifikacijos paprastumą, aiškumą ir suprantamumą. Ši savybė siejasi su specifikacijos konceptualumu.
  - Jei reikalavimai saugomi duomenų bazėje ir tvarkomi programiškai, koncepcinė skaidra dažniausiai yra paaukojama siekiant padidinti reikalavimų apdorojimo našumą.
- ***Konkretumas*** *(constructability)***:** specifikacija yra konkreti, jei gali būti patikrintas visų joje suformuluotų reikalavimų įgyvendinimo laipsnis.
  - Tai labai svarbi specifikacijos savybė, nes šis dokumentas yra užsakovo ir vykdytojo sandorio dalis ir jos pagrindu yra sprendžiama ar sandoris buvo įvykdytas.
- ***Geras struktūrizavimas:*** joje griežtai išlaikytas turinių atskyrimo principas.
- ***Tikslumas:*** visi joje suformuluoti reikalavimai yra tikslūs.
  - Tai labai svarbi specifikacijos savybė, nes šis dokumentas yra užsakovo ir vykdytojo sandorio dalis ir jos pagrindu yra sprendžiama ar sandoris buvo įvykdytas.

- ***Išsamumas:*** specifikacijoje aprašytas visas reikalingas sistemos funkcionalumas ir visi joje pateikti reikalavimai yra išsamūs.
  - Tai reiškia, kad specifikacijoje yra viskas, kas joje turėtų būti.
    - Patikrinti, ar specifikacija tikrai yra išsami, yra labai sunku.
    - Nagrinėjant tai, kas yra aprašyta, labai sunku pastebėti, ko gi ten trūksta.
    - Geriausias būdas išsamumui patikrinti yra sukurti sistemos prototipą.
  - Išsamioje specifikacijoje turi būti aprašyta, kaip sistema reaguoja į kiekvieną iš galimų įvesties tipų kiekvienoje galimoje situacijoje.
    - Reikia nagrinėti ne tik kaip reaguoja sistema į leistinas (teisingas) įvestis, bet ir kaip ji reaguoja į klaidingas įvestis.
  - Išsamioje specifikacijoje turi būti visos reikalaujamos to dokumento dalys, visi puslapiai, visi paveikslėliai ir visos lentelės turi būti sunumeruoti, paveikslėliai ir lentelės turi turėti pavadinimus, turi būti pateiktos tvarkingos nuorodos į visus naudojamus išorinius informacijos šaltinius.
    - Tai formalus dokumento išsamumas, kurį galima vadinti ir dokumento tvarkingumu.
    - Kokias dalis turi turėti dokumentas, nustato vidiniai organizacijos (arba projekto) standartai.
  - Kartais analizės metu visų reikalavimų specifikuoti nepavyksta, kai kurių reikalavimų specifikavimą tenka atidėti vėlesniam laikui. Tokie reikalavimai pažymimi žyme AVL.
    - Išsamioje specifikacijoje kiekvienam AVL žyme pažymėtam reikalavimui turi būti nurodyta:
      - kodėl reikalavimo formulavimas yra atidėtas;
      - kas turi būti atlikta, kad reikalavimą būtų galima suformuluoti;
      - iki kada reikalavimas turi būti suformuluotas;
      - kas atsakingas už tai, kad reikalavimas būtų suformuluotas nurodytu laiku.
- ***Vienareikšmiškumas:*** joje neturi būti jokių dviprasmybių.
  - Tai labai svarbi specifikacijos savybė, nes šis dokumentas yra užsakovo ir vykdytojo sandorio dalis ir jos pagrindu yra sprendžiama ar sandoris buvo įvykdytas.
- ***Trasuojamumas:*** reikalavimai yra lokalizuojami ir reikalavimų ir projektinę specifikacijas galima tarpusavyje susieti kryžminėmis nuorodomis.
  - Jei visi reikalavimai yra sunumeruoti, į juos galima daryti nuorodas.
- ***Darnumas:*** visi joje suformuluoti reikalavimai yra integruojami, jokių prieštaravimų dokumente nėra.
  - Nedarna gali atsirasti dėl įvairių priežasčių, pavyzdžiui, dėl:
    - terminų konflikto: skirtingose dokumento vietose tas pats daiktas yra vadinamas skirtingai;
    - savybių konflikto: skirtinguose reikalavimuose yra reikalaujama, kad sistema turėtų kokias nors nesuderinamas savybes;
    - naudojimo režimų konflikto: pavyzdžiui, vienoje vietoje reikalaujama, kad su sistema būtų dirbama dialogo režimu, kitoje, kad ji būtų sistema, veikianti paketinio duomenų apdorojimo režimu.
- ***Keičiamumas:*** dokumentas turi būti lengvai keičiamas.
  - Specifikaciją lengva keisti, jei ji parašyta griežtai prisilaikant turinių atskyrimo principo ir visi reikalavimai turi unikalius numerius.
- ***Naudojimo patogumas:*** mažai kas skaito visą dokumentą, kiekvienam reikia tik to, kas jam svarbu. Todėl dokumentas turi būti parašytas taip, kad juo būtų galima naudotis kaip žinynu.
- Reikalavimų specifikacija pateikiama varžovams besivaržantiems konkurse dėl projekto.
  - Tinkamai organizuotas konkursas padeda užsakovui pasirinkti vykdytoją, kuris reikalavimus įgyvendins pigiausiai.
- Kad konkurse galėtų dalyvauti kuo daugiau pretendentų, reikalavimai turi būti suformuluoti kuo bendriau.
  - Kita vertus, reikalavimai turi būti pakankamai konkretūs, kad konkurse negalėtų dalyvauti pretendentai, siūlantys sukurti ne tai, ko iš tiesų reikia užsakovui.
  - Idealiu atveju, reikalavimai turėtų būti suformuluoti taip, kad visi galimi pasiūlymai būtų suskirstyti į dvi dalis: tuos, kurie tenkina vartotojų poreikius, ir tuos, kurie tų poreikių netenkina.

10 paskaita

Kokie yra dalykinės srities analizės tikslai? Kas lemia dalykinės srities analizės svarbą? (2)
Tikslai:

1. Informacijos apie dalykinę sritį surinkimas
2. Klasifikuoti ir apibendrinti faktus
3. Specifikuoti reikalavimus
4. Vertinti reikalavimus

Dalykinės srities analizės svarba lemia:

1. Tai, kad ši analizė yra pirmoji informacinių sistemų kūrimo stadija
2. Pirmoji verslo sistemų reinžinerijos ir kokybės valdymo procesų stadija
3. Dalykinės srities analizė turi būti atliekama prieš pradedant bet kuriuos organizacijos pertvarkymus, nepaisant to, kokiame lygmenyje ir dėl kokių priežasčių jie yra daromi
4. Daugelyje organizacijų apie 40-60 procentų IT biudžeto sunaudojama programų sistemų priežiūros ir tobulinimo darbams. Didžioji dauguma šių darbų, atsiranda dėl neišsamios ar blogai atliktos dalykinės srities analizės, atliktos kuriant sistemą.

Kokia yra dalykinės srities analizės iš viršaus žemyn metodo esmė? (2)
Kuriant sistemą "iš viršaus žemyn" būdu, pirmose stadijose turi būti samprotaujama viršutinių abstrakcijos lygmenų sąvokomis. Tai reiškia, kad nagrinėjami apibendrinti (agreguoti) procesai, funkcijos ir duomenys, o ne detalios užduotys. Vietoje to, kad pradėti gilintis į konkrečių užduočių detales, analitikas privalo susidaryti mažiau detalų (nors nebūtinai mažiau sudėtingą) bendrą visos organizacijos vaizdą ir, suprantant organizacijos verslą vis geriau ir geriau, vis labiau ir labiau detalizuoti tą vaizdą.

Koks vaidmuo jam tenka, atliekant dalykinę analizę? (3)
Analizės metu, o dažnai ir iki projektavimo pabaigos darbą su personalu, resursų skirstymą, planavimą, progreso stebėjimą, projekto prezentacijas, techninį vadovavimą analizės metu, vykdo analitikas. Analitiko vaidmenys atliekant analizę:

1. Reporteris.
2. Detektyvas.
3. Konsultantas.
4. Diagnostikas.
5. Tyrėjas.
6. Organizatorius
7. Galvosūkių sprendėjas
8. Vertintojas
9. Paprastintojas
10. Genties žvalgas
11. Menininkas
12. Skulptorius

Kokios yra svarbiausios sisteminio analitiko užduotys? (3)
Sisteminio analitiko užduotys:

1. Pradinėse projekto stadijose nurodyti veiklos kryptis vykdytojų kolektyvui.
2. Išdetalizuoti "verslo žinias".
3. Perprasti vartotojų problemas ir pažvelgti į jas vartotojų akimis.
4. Išryškinti ir suformuluoti verslo problemas.
5. Pasiūlyti kaip praktiškai ir kuo pigiau spręsti vartotojus dominančias verslo problemas.
6. Pasiūlyti verslo problemų sprendimo būdų palaikymo IT priemonėmis alternatyvas.
7. Prieš pradedant sistemos realizavimą, išnagrinėti sistemos funkcijų įgyvendinamumo ir sistemos įdiegiamumo klausimus ir pasiūlyti, kaip spręsti galimas problemas.
8. Išsiaiškinti, ką dar reikia papildomai panagrinėti, atlikti tokius nagrinėjimus ir dokumentuoti jų rezultatus.
9. Sekti, kad vykdytojų kolektyvas nenukryptų nuo projekto tikslų.

10. Parengti tiksliai ir suprantamai dokumentuoti operacinius vartotojų poreikius.
11. Perteikti PS inžinieriams informaciją apie vartotojų poreikius ir specifikuoti kuriamos sistemos reikalavimus.
12. Pristatyti projektą užsakovui ir savo vadovybei..
13. Sekti, kad vykdant projektą būtų laikomasi organizacijos standartų ir procedūrų.
14. Sekti, kad projekto rezultatai tenkintų sandorio ir kitus privalomus reikalavimus

Ką šiandien apima duomenų apdorojimo samprata? (2)
Nors tradiciškai terminas "duomenų apdorojimas" apima ir rankinį duomenų apdorojimą, šiandien pagrindinis dėmesys čia skiriamas atitinkamų programų sistemų kūrimui, diegimui, naudojimui, aptarnavimui ir priežiūrai. Anksčiau programų sistemos dažniausiai buvo skirtos organizacijos valdymo funkcijoms kompiuterizuoti. Paskutiniu metu dėmesio centre atsidūrė sistemos, skirtos operacinio lygmens funkcijoms kompiuterizuoti. Šiandien organizacijoje jau sunku rasti tokį veiklos barą, kuris nebūtų daugiau ar mažiau kompiuterizuotas.

Kaip suprantama dalykinės srities funkcinė analizė? Kokie yra svarbiausi funkcinės analizės tikslai? (4)
Dalykinės srities funkcinė analizė aprašo vartotojo funkcijų sąveiką su kitų veiklos sričių funkcijomis. Tikslai:

1. Nustatyti organizacinę struktūros schemą
2. Aprašyti, kokią vietą būsimieji sistemos vartotojai ir padaliniai, kuriuose jie dirba, užima organizacijos struktūroje
3. Sudaryti verslo pobūdžio aprašą
4. Būsimųjų sistemos vartotojų funkcijų ir jų atliekamo darbo aprašas;
5. Specifinių kiekvieno vartotojo funkcijų aprašai;
6. Problemų, su kuriomis vartotojai susiduria savo darbe, aprašas
7. Tų problemų poveikio vartotojų atliekamam darbui ir visos organizacijos veiklai aprašas;
8. Funkcinių ribojimų aprašas
9. Nustatyti funkcinę vartotojų priklausomybę nuo kitų darbuotojų;
10. Nustatyti vartotojų finansinės ir kitokios kontrolės procedūras.

Kaip suprantama dalykinės srities procesų analizė? Kokie yra svarbiausi procesų analizės tikslai? (4)
Funkcinės analizės metu identifikuotoms ir aprašytoms funkcijoms, procesų analizės metu išsiaiškinama, kokie verslo procesai ir kokios veiklos yra vykdomi toms funkcijoms realizuoti. Tikslai:

1. Aprašyti visus procesus ir visas veiklas, kuriuos vykdant dalyvauja būsimieji kuriamos sistemos vartotojai ir nurodyti, kuriuos iš tų procesų bei veiklų verta kompiuterizuoti, kodėl ir kokiu mastu tai tikslinga daryti
2. Išsiaiškinti ir aprašyti kaip kiekvienos verslo funkcijos procesai ir veiklos yra susiję su kitų verslo funkcijų procesais ir veiklomis
3. Kiekvienam procesui ir kiekvienai veiklai aprašyti verslo transakcijas, inicijuojančias tuos procesus bei veiklas
4. Nustatyti kas vykdo kiekvieną procesą ir kiekvieną veiklą
5. Nustatyti kokius, kada ir kodėl atlieka veiksmus ir kokios bus pasekmės jeigu tie veiksmai nebus atliekami (arba atliekami neteisingai)

Kaip suprantama dalykinės srities užduočių analizė? Kokie yra svarbiausi užduočių analizės tikslai? (4)
Užduotis yra smulkiausia dalykinės srities analizės metu analizuojama verslo funkcijos dalis. Užduočių analizės metu nusileidžiama į patį žemiausią verslo sistemos dekompozicijos lygmenį. Paprastai, užduočių analizė nėra traktuojama kaip savarankiškas analizės stadijos etapas. Ji yra perpinama su procesų analize. Tikslai:

1. Nustatyti kiekvienos užduoties įeigą, išeigą ir apdorojančią dalį
2. Nustatyti kiekvieną būsimos sistemos vartotojų vykdomą užduotį ir išsiaiškinti kaip, kada, kodėl ir prie kokių sąlygų ji yra vykdoma ir ar nėra geresnių būdų jas vykdyti.
3. Nustatyti kaip galima užduotį supaprastinti.

4. Nustatyti ar užduoties vykdymą reglamentuoja kokie nors standartai ar kokios nors nustatytos procedūros
5. Nustatyti ar tą užduotį vykdo tik tas darbuotojas, ar ją arba labai panašias į ją užduotis vykdo ir kokie nors kiti darbuotojai
6. Nustatyti kaip dažnai, vykdant užduotį, susidaro kokios nors ypatingos situacijos ir kaip jos yra apdorojamos. Kiek laiko paprastai prireikia susidoroti su tokiomis situacijomis?
7. Nustatyti ar galima kaip nors pakeisti verslo procesą ar padaryti kažką kitą, kad ypatingos situacijos išnyktų arba kad jų bent jau ženkliai sumažėtų
8. Nustatyti ar užduoties vykdyme yra kokių nors ciklų arba išsišakojimų
9. Nustatyti kiek vidutiniškai ilgai užtrunka vienas užduoties vykdymas
10. Nustatyti ar užduočiai vykdyti žmogus turi būti kaip nors specialiai apmokomas arba treniruojamas

Kaip suprantama dalykinės srities duomenų analizė? Kokie yra svarbiausi duomenų analizės tikslai? (4)

Duomenų analizės užduotis yra, duomuo po duomens, suformuluoti duomenų reikalavimus. Kiekvienas duomuo yra apibrėžiamas vadovaujantis verslo poreikiais, nustatoma, kas yra jo savininkas, kas juo naudojasi ir iš kur jis yra gaunamas. Duomenys yra grupuojami į įrašus ir galų gale visi verslo duomenys vienaip ar kitaip sudėliojami į kokias nors duomenų struktūras.

Tikslai:

1. Nustatyti kokie duomenys naudojami dabar
2. Nustatyti kokių duomenų reikės sukūrus sistemą
3. Išanalizuoti duomenų ir informacijos perdavimo būdus ir maršrutus organizacijos viduje
4. Nustatyti organizacijoje naudojamus dokumentus ir formas

Kokiu tikslu imamas interviu? (2)

Interviu imamas tikslu išsiaiškinti, kaip interviu duodantis asmuo žiūri į vieną ar kitą savo veiklos aspektą, ką jis mano analitikui rūpimu klausimu. Interviu primena pasitarimą ar posėdį, bet jame dalyvauja tik du asmenys: imantysis interviu ir duodantysis interviu. Gali dalyvauti ir keli imantys/duodantys interviu, bet turi būti tik vienas pagrindinis imantis/ duodantis interviu.

Apibūdinkite pagrindinį interviu? (2)

Surinkti informaciją reikalingą vartotojų poreikiams nustatyti ir sistemos reikalavimams suformuluoti:

1. interviu faktams, nuomonėms ir kitai informacijai apie verslo sistemą surinkti;
2. interviu surinktai informacijai patikrinti (kai kurie surinkti faktai gali būti klaidingi, apklausiami kiti žmonės);
3. interviu įsitikinti, kad analitikas teisingai interpretuoja surinktą informaciją;
4. interviu surinktai informacijai papildyti ir patikslinti (vykdomi po pirmo interviu informacijos apdorojimo);

Iš kokių komponentų susideda interviu? (2)

Interviu komponentai:

1. Apklausiamųjų parinkimas ir interviu laiko derinimas.
2. Interviu klausimyno rengimas.
3. Interviu ėmimas.
4. Interviu dokumentavimas.
5. Interviu teksto peržiūra kartu su interviu davusiu asmeniu.
6. Galutinio interviu teksto rengimas.

Kokiais tikslais yra imami interviu? (2)

Interviu ėmimo tikslai:

1. surinkti informaciją apie įmonę, verslo funkcijas, verslo procesus, veiklas, užduotis ir duomenis;
2. nustatyti verslo problemas ir jų sprendimo būdus;
3. nustatyti operacinius vartotojų poreikius;
4. patikrinti, patikslinti ar papildyti anksčiau surinktą informaciją;
5. išsiaiškinti sistema suinteresuotų asmenų nuomonę kokiu nors klausimu;
6. pasirengti kitiems interviu.

Kokiomis taisyklėmis patartina vadovautis rengiant ir imant interviu? (4)
Interviu metu vyksta informacijos mainai tarp dviejų asmenų, todėl svarbu jį organizuoti taip, kad tą tikslą tikrai pasiekti. Kadangi interviu nėra reklamos akcija, reikia sukurti tam tinkamą psichologinę atmosferą. Interviu procesas yra vienas iš interpersonalinio bendravimo procesų, tačiau jis tuo pat metu yra ir tyrimo bei problemų sprendimo procesu. Taigi, tai yra žaidimas ir interviu reglamentuojančios taisyklės iš tiesų yra tam tikro žaidimo taisyklės.

Rekomendacijos:

1. Draugišką atmosferą.
2. Paaiškinti apklausiamajam, kodėl toks interviu reikalingas ir kodėl nutarta imti interviu būtent iš jo.
3. Pabrėžti interviu metu gautos informacijos svarbą verslo problemoms spręsti ir paaiškinti, kaip konkrečiai ta informacija bus panaudota.
4. Įgyti apklausiamojo pasitikėjimą, priversti jį jaustis suinteresuotu asmeniu ir aktyviai dalyvauti interviu..
5. Garantuoti pokalbio konfidencialumą ir užtikrinti, kad pokalbio metu surinkta informacija bus išviešinta tik po to, kai jis ją peržiūrės ir leis ją paviešinti.
6. Užtikrinti, kad interviu neturėtų jokių neigiamų pasekmių apklausiamajam.
7. Neturėti jokių išankstinių nuostatų tuo klausimu, kuriuo renkama informacija.
8. Nesistengti primesti apklausiamajam savo nuomonės.
9. Klausimus formuluoti taip, kad jie prasidėtų klausiamaisiais žodeliais, t.y. žodeliais kas, kuris, kur, kada, kodėl, kaip.
10. Nereikalauti, kad būtinai atsakytų tik taip arba ne.
11. Užduodant papildomus klausimus, pasitikrinti, ar teisingai suprantama, kas yra sakoma..
12. Nesiginčyti su apklausiamuoju.
13. Stengtis nenukrypti nuo interviu temos ir nuo klausimyne numatytų klausimų.
14. Nepertraukinėti kalbančiojo.
15. Neparodyti kalbančiajam, kad kas nors yra neįdomu arba jau žinoma..
16. Fiksuoti, kas yra sakoma, bet tai daryti taip, kad kalbančiajam nekiltų kokių nors sunkumų kalbėti.
17. Kalba duodantysis interviu, imantysis interviu klauso.
18. Kaip galima greičiau dokumentuoti interviu.
19. Planuoti interviu trukmę ir griežtai jos laikytis.
20. Neprieštarauti, jei duodantysis interviu panorės vėliau ką nors keisti ar tikslinti.
21. Būti mandagiu ir nuoširdžiu.

Iš ko rekomenduojama imti interviu? (4)
Atsakant į šį klausimą, reikia pasirinkti apklausiamuosius asmenis, suprasti, ko iš kiekvieno iš tų asmenų galima tikėtis, kaip su jais bendrauti, kaip patikrinti iš jų gautą informaciją ir, svarbiausia, suprasti, koks yra jų požiūris į vykdomąjį projektą. Paprastai užsakovas kontaktams su vykdytojais skiria savo įgaliotąjį atstovą. Šis atstovas turi padėti nuspręsti, su kuo reikia kalbėtis vienu ar kitu klausimu, ir pristatyti apklausiamiesiems analitikus. Jis taip pat gali padėti analitikams interpretuoti surinktą informaciją. Paprastai analitikas turi galimybe apklausti bet kurį įmonės darbuotoją, tačiau nėra jokios prasmės apklausti visus darbuotojus, ypač tais atvejais, kai įmonė yra pakankamai didelė. Apskritai, asmenys, iš kurių imami interviu, gali būti suskirstyti į tris grupes:

1. įmonės vadovai,
2. padalinių ir/arba veiklos barų vadovai,
3. eiliniai įmonės padalinių darbuotojai.

Įmonės vadovai:

1. Pokalbius patariama pradėti nuo įmonės vadovo, nes būtent vadovas geriausiai gali apžvelgti visą įmonės veiklą ir jos verslo funkcijas ir nusakyti visą kompiuterizuojamos veiklos kontekstą
2. Vadovas gali suformuluoti pagrindinius kuriamos sistemos tikslus, paaiškinti, ko iš jos tikimasi, pasakyti, kokius resursus numatoma skirti projektui ir kada tikimasi pradėti eksploatuoti sistemą.
3. Jis taip pat gali patarti, su kokiais darbuotojais verta kalbėtis ir su kokiais kitais informacijos šaltiniais reikėtų susipažinti.
4. Be to, vadovo nuomonė yra svarbiausia, sprendžiant kokią verslo tobulinimo strategiją geriausiai pasirinkti.

5. Įmonės vadovas gali pateikti analitikui įmonės organizacinės struktūros schemą, papasakoti, kiek žmonių kokiame padalinyje dirba, ir aptarti tų padalinių funkcijas. Ši informacija yra labai vertinga planuojant tolimesnę įmonės darbuotojų apklausą
6. Analitikas neturėtų tikėtis gauti iš įmonės vadovo labai detalią ir išsamią informaciją. Vadovas nežino ir negali žinoti visų verslo sistemos detalių ir gali labai nedaug ką pasakyti apie tai, kaip vykdomos konkrečios užduotys.
7. Be to, nereikia pamiršti, kad būtent vadovas yra labiausiai suinteresuotas projekto sėkme, galbūt pats tą projektą iniciavo ir galų gale tars galutinį žodį, ar liko patenkintas projekto rezultatais.
8. Todėl būtina gerai suprasti, ko gi šis asmuo iš jūsų tikisi!

Žemesniųjų lygmenų vadovai:

1. Įmonėje gali būti daug valdymo lygmenų, kiekvienas žemesnio lygmens vadovas turi savo atsakomybės barą ir juo tas lygmuo yra žemesnis, tuo ta atsakomybė yra konkretesnė ir griežčiau apibrėžta.
2. Pasirenkant ką reikėtų apklausti iš žemesniųjų lygmenų vadovų, į sąrašą patartina įtraukti

(a) visus įmonės vadovo pavaduotojus; (b) visus vadovus, kurių vadovaujamos veiklos barą vienaip ar kitaip palies kuriamoji sistema.To reikia bent jau tam, kad išsiaiškinti su kokiais kitais jų vadovaujamais darbuotojais reikėtų pasikalbėti.

1. Tokių pokalbių metu gali paaiškėti, kad analitiko prielaidos apie verslo problemų priežastis yra neteisingos ir kad analizės sferą reikia gerokai išplėsti..
2. Todėl, kalbantis su įmonės vadovu, reikėtų iš anksto paprašyti leidimo kalbėtis ir su tais padaliniais, kurie iš pirmo žvilgsnio atrodo neturintys nieko bendro su nagrinėjamomis verslo problemomis.
3. Vadovo taip pat reikėtų paprašyti, kad jis visus žemesniųjų lygmenų vadovus supažindintų su projekto tikslais ir perspėtų, kad analitikams gali prireikti pasikalbėti su jais pačiais arba su jų vadovaujamais žmonėmis.
4. Neretai darbuotojai kartas nuo karto keičia savo pareigybes ir netgi savo veiklos įmonėje pobūdį. Tokie darbuotojai yra labiau patyrę ir, pasirinkus juos apklausai, iš jų galima gauti vertingesnės informacijos.
5. Vadovas, kuris palaipsniui kilo iš vieno lygmens į kitą, gali apžvelgti tam tikrą veiklos barą iš skirtingų valdymo lygmenų perspektyvos ir, netgi jei nebežino konkrečių užduočių vykdymo dabartiniu laiko momentu detalių, gali papasakoti daug daugiau, negu tas, kuris visą laiką dirbo tą patį darbą. Būtent tokie žmonės gali daugiausiai pasakyti apie stebimų verslo problemų priežastis ir patarti, kaip jas pašalinti
6. Žemesniųjų lygmenų vadovai taip pat gali padėti geriau suprasti aukštesniųjų lygmenų vadovų pateiktą informaciją ir tų vadovų intencijas.
7. Analitikas privalo suprasti, kad darbuotojo požiūrį į nagrinėjamas problemas lemia jo vaidmuo įmonėje ir kad nei vienas iš tų požiūrių nėra ir negali būti objektyvus. Todėl problemas reikia "apžiūrėti iš visų pusių". Kiekvienas požiūris padeda analitikui priartėti prie objektyvesnio tikrosios padėties vertinimo.
8. Viena iš svarbiausių analitiko apklausos meto užduočių yra tikrinti visus faktus ir nuomones, apklausiant skirtinguose vaidmenyse esančius asmenis. Kryžminės apklausos turi būti maksimaliai objektyvizuotos, apklausiamieji neturi sužinoti, ką vieni iš jų mano apie kitų darbą. Ne analitiko reikalas ieškoti kaltųjų ir rodyti pirštu į tuos, kurie ką nors ne taip daro. Jam reikia tik objektyviai įvertinti situaciją. Gavus prieštaringą informaciją, reikia ieškoti kokio nors trečio šaltinio

Eiliniai įmonės padalinių darbuotojai:

1. Analitikui reikia nuspręsti, kaip būsimoji sistema padės jiems jų darbe, o tam reikia išsiaiškinti, ką jie daro šiuo metu bei kodėl ir kaip jie tą daro.
2. Didžioji dauguma interviu turi būti imama būtent iš eilinių darbuotojų ir būtent šie interviu turi būti detaliausi.
3. Reikia parinkti apklausai bent po vieną kiekvienos kompiuterizuojamos užduoties vykdytoją. Kartais verta apklausti ir daugiau vykdytojų. Kiekvienas naujas apklaustasis gali pasakyti ką nors, ką praleido ar laikė nesvarbiu kiti. Tokiais atvejais reikia patikrinti, ar anksčiau apklaustieji sutinka su tuo, ką papildomo pasakė naujas apklaustasis.

4. Eilinių darbuotojų pateikta informacija turi būti sugretinta su jų tiesioginių vadovų pateikta informacija ir analitikas privalo eliminuoti visus šitaip išryškintus prieštaravimus.

Kodėl būtina dokumentuoti interviu? (4)

Dokumentuoti interviu būtina ir dėl semantinių priežasčių, ir dėl žmogiškosios atminties savybių. Pokalbis pasimiršta, tai, kas buvo pasakyta, galima suprasti skirtingai. Tikslumas ir perteikimas:

1. Rašytinis dokumentas yra tikslesnis, jį galima peržiūrėti kelis kartus, kol nebus susitarta dėl to, kaip ką reikia pasakyti.
2. Jis turi privalumą, kad jame galima daryti nedidelius lokalius pakeitimus, nekeičiant pagrindinės dokumento minties, jo bazinių nuostatų.
3. Be to, raštu išdėstytą idėją bet kuriuo momentu galima su visomis jos detalėmis atkurti atmintyje ir perteikti kitiems asmenims.
4. Kita vertus, dokumento negalima nieko paklausti, todėl rašytinis tekstas turi neturėti jokių neaiškumų ar dviprasmybių.

Grafika:

1. Dokumentuose pateikiami ne tik tekstas, bet ir įvairi iliustracinė medžiaga.
2. Ši medžiaga papildo ir paaiškina tekstą. Kartais geras piešinys atstoja tūkstantį žodžių!
3. Paveikslėliai ir grafika turi tą pranašumą, kad, skirtingai negu tekstas, jie iš karto pateikia visumą, o ne atskiras jos detales

Dokumentavimas padeda geriau suprasti mintį, bet, kas yra dar svarbiau, padeda dirbti su analizės rezultatais. Analitikas sukuria įrašus, į kuriuos vėliau galima daryti nuorodas ir kurių pagrindu priimami tolimesni sprendimai. Gera dokumentacija eliminuoja poreikį iš naujo klausinėti apklaustuosius to, ko jie jau buvo klausti. Dokumentavimas yra nuobodus ir varginantis darbas. Tačiau jis yra būtinas: neturint dokumentuotų interviu ir vieną analitiką pakeitus kitu, visą analizę reikėtų pradėti iš naujo. Be analizės dokumentų, negalima būtų parengti sistemos reikalavimų specifikacijos ir pradėti projektuoti sistemą.

Koks interviu vadinamas telefoniniu? Kokie yra tokio interviu privalumai ir trūkumai? (3)

Telefoniniai interviu: naudojami renkant atsakymus į klausimus, į kuriuos galima atsakyti “taip” arba “ne”.

Ypatumai:

1. Galima apklausti tik santykinai nedidelį respondentų skaičių.
2. Pavyksta apklausti daugumą respondentų.
3. Apklausa gali būti atlikta per pakankamai nedidelį laiką.
4. Vidutinio dydžio apklausos kaštai.
5. Tinka santykinai paprastai informacijai rinkti.

Privalumai:

1. Imantysis interviu gali užduoti papildomus klausimus
2. Tai greitesnis ir patikimesnis metodas negu apklausa kompiuteriniu paštu
3. Tai pigiausias interviu ėmimo būdas

Trūkumai:

1. Sunku sukurti tinkamą psichologinę atmosferą
2. Tenka labiau riboti aptariamų klausimų ratą
3. Pavargstama greičiau negu bendraujant žodžiu
4. Sunku planuoti interviu trukmę
5. Galimi ryšio trykiai

Kokią informaciją galima rinkti paštu? Kokie yra tokio informacijos rinkimo būdo privalumai ir trūkumai (3)

Informacijos rinkimas paštu: naudojamas, kuomet reikia surinkti iš didelio skaičiaus asmenų daug, kiekybinių duomenų. Gali būti renkama tiek kompiuteriniu, tiek paprastu paštu. Ypatumai

1. Galima rinkti informaciją iš didelio asmenų skaičiaus.
2. Apklausa dažniausiai užtrunka gana ilgai.

3. Nėra galimybių iš karto užduoti papildomus klausimus.
4. Maži apklausos kaštai.
5. Geriausiai tinka kiekybinio pobūdžio duomenims rinkti

Privalumai:

1. Į klausimus galima atsakinėti atsakinėjančiajam patogiu laiku ir jam patogioje vietoje.
2. Klausimai suformuluoti raštu, kas palengvina jų supratimą
3. Galima apklausti didelį asmenų skaičių
4. Nesunku administruoti ir apdoroti

Trūkumai:

1. Sunku prognozuoti, kiek asmenų atsakys į pateiktus klausimus
2. Su apklausiamuoju nėra tiesiogiai bendraujama, dėl to negalima stebėti jo reakcijos į pateiktus klausimus
3. 30% - 40% atsakiusiųjų yra labai geras rezultatas. Paprastai atsako tik apie 10% respondentų
4. Sunku surinkti išsamią informaciją
5. Sunku patikslinti neaiškius atsakymus

Kas tai yra fokuso grupė? Kokie yra fokuso grupių privalumai ir trūkumai? (3)
Fokuso grupės: naudojamos gauti skirtingas reakcijas į kokį nors vieną klausimą. Jei visi grupės dalyviai yra panašios kvalifikacijos ir panašaus statuso, jose dažnai pasireiškia sinergetinis efektas. Ypatumai

1. Nedidelis respondentų skaičius.
2. Palyginti aktyvus respondentų dalyvavimas apklausoje.
3. Apklausa atliekama gana per trumpą laiką.
4. Gana aukšti apklausos kaštai.
5. Gerai tinka nuomonėms, o taip pat paprastiems ar labai išdetalizuotiems duomenims rinkti.

Privalumai:

1. Jei atsakymai nėra pakankamai aiškūs, juos lengva patikslinti.
2. Tuo pat metu renkami duomenys iš daugelio šaltinių.
3. Gaunama informacija automatiškai patikrinama diskusijų metu.

Trūkumai:

1. Jei pasitarimas nėra tinkamai vedamas, grupė lengvai nukrypsta nuo svarstomo klausimo.

Kam naudojami apsilankymai darbo vietose? Kokie yra šio metodo privalumai ir trūkumai? (3)
Apsilankymai darbo vietose: naudojamas, norint tiesiogiai susipažinti su darbo procesais, užduočių vykdymu, darbo aplinka ir darbo sąlygomis. Ypatumai:

1. Galima aplankyti tik nedaugelį darbo vietų.
2. Informacijos rinkimas užtrunka gana ilgai.
3. Dideli kaštai.
4. Naudotinas, kai reikia išsiaiškinti kitais metodais sunkiai išsiaiškinamus niuansus.

Privalumai:

1. Pagerina analitiko turimą kompiuterizuojamos veiklos sampratą.
2. Gaunama informacijos, kurios negalima surinkti kitais būdais.
3. Galima patikrinti kitais būdais surinktą informaciją.

Trūkumai:

1. Nesąžiningi ar nekompetetingi darbuotojai apsilankymu gali pasinaudoti kaip ekskursija ir jokios vertingos informacijos nesurinkti.
2. Galima aplankyti tik nedidelį darbo vietų skaičių.
3. Didelės laiko ir kitų resursų sąnaudos.
4. Apsilankymo planą reikia parengti labai iš anksto, nes jį reikia suderinti.
5. Didelė tikimybė, kad dėl nenumatytų aplinkybių planas sužlugs.

Paaiškinkite, kas tai yra kolektyvinis projektavimas ir kolektyvinė reikalavimų analizė. (3)
Anksčiau vyravusią praktiką, kuomet vykdytojas pasiųsdavo pas užsakovą analitiką ar analitikų grupę paimti interviu pas dalykinės srities specialistus, šiandien dažnai išstumia kitoks darbo stilius: dalykinės srities specialistai yra suburiami į vieną grupę, vadovaujamą analitiko-moderatoriaus, ir patys analizuoja verslo funkcijas, procesus, veiklas bei duomenis. Tai ilgalaikės grupės. Kolektyvinio darbo metodas yra grindžiamas prielaida, kad dalykinės srities specialistai, vadovaujami patyrusio moderatoriaus, gali išsiaiškinti savo operacinius poreikius pakankamai detaliai, kad iš juos būtų galima transformuoti į programų sistemos reikalavimų specifikaciją.

Paaiškinkite, kokie yra PS prototipų naudojimo privalumai ir trūkumai. (3)
Nors kaip prototipą galima panaudoti bet kurį sistemos modelį, paprastai prototipu vadinamas modelis, modeliuojantis vartotojo interfeiso ekranus, užklausas bei generuojamas ataskaitas. Prototipai dažniausiai naudojami vertinti sistemos patogumą vartotojams. Prototipas gali būti veikiantis arba ne**.** Paprastai prototipas neatlieka jokių skaičiavimų. Vartotojas gali matyti kaip sistemoje atrodys interfeisai, vesti duomenis, formuluoti užduotis, matyti, kokias ekranines ataskaitas generuos sistema, bet negali gauti jokių realių rezultatų.

Privalumai:

1. Leidžia įtraukti vartotoją į sistemos projektavimo ir vertinimo procesus.
2. Vartotojas gali "pasižaisti" su interfeisu dar prieš pradedant realizuoti sistemą.
3. Prototipas sukuriamas labai greitai.
4. Jis kainuoja labai mažai
5. Prototipas lengvai modifikuojamas.
6. Nepasiteisinus, jį, be didelių nuostolių, galima išmesti.

Trūkumai:

1. Niekaip neimituoja atliekamų skaičiavimų.
2. Kaip taisyklė, prototipas negali pasinaudoti nei realiais, nei testavimo duomenimis ir parodyti vartotojui, kaip jie atrodo.
3. Kartais ir vartotojai ir netgi programuotojai pradeda traktuoti prototipą kaip galutinio produkto užuomazgą.
4. Dėmesys sutelkiamas į interfeisą, pamirštami algoritminiai sistemos aspektai.
5. Kadangi dažniausiai prototipų generatoriai neturi galimybių realizuoti skaičiavimus, realią sistemą tenka pradėti kurti nuo nulio su kita instrumentine sistema.
6. Nesant skaičiavimų, vartotojas neturi aiškaus įsivaizdavimo, kaip iš tiesų dirbs reali sistema.
7. Pridedant skaičiavimus, gali prireikti esminiai pakeisti interfeisus ir visas atliktas darbas gali prarasti bet kokią prasmę.

Kokiu tikslu atliekamas reikalavimų vertinimas? (2)
Vertinimas atliekamas siekiant įsitikinti, kad:

1. visos šalys sutinka su suformuluotais teiginiais;
2. visi suformuluoti teiginiai yra išsamūs, korektiški ir neturi jokių dviprasmybių.
3. visi analizės rezultatai yra pagrįsti analizės metu surinkta informacija.
4. sistema spręs realias verslo problemas ir tenkins realius dalykinės srities specialistų poreikius

Paaiškinkite, kokiu tikslu yra atliekama darnos analizė? (4)

Duomenų darnos analizė atliekama siekiant užtikrinti, kad:

1. visi duomenys yra taisyklingai apibrėžti ir turi tinkamus pavadinimus;
2. visos duomenų transformacijos yra žinomos ir dokumentuotos;
3. yra žinomos ir dokumentuotos visos duomenų pateikties formos;
4. visos duomenų ir jų transformacijų apibrėžtys yra korektiškos.
5. yra žinomi ir dokumentuoti visi išskaičiuojami duomenys ir jų skaičiavimo būdai.

Užduočių darnos analizė atliekama siekiant užtikrinti, kad:

1. visos programų sistemos užduotys tarpusavyje yra susietos teisingai;
2. visos programų sistemos užduotys bus įvykdomos per reikiamą laiką;
3. vykdant visas užduotys bus prieinami joms vykdyti reikalingi resursai;

4. visos programų sistemos užduotys yra suderintos su atitinkamomis verslo užduotimis ir pareigybinėmis instrukcijomis;
5. asmenys, kurie, kaip manoma vykdys užduotis, turi tam reikiamus įgaliojimus;
6. vadovybė supranta programų sistemos užduotis taip pat, kaip ir asmenys, kurie jas vykdys.

Lygmenų darna:

1. Dideliuose projektuose gali būti keletas analizės lygmenų.
2. Reikia patikrinti ar analizės lygmenys tikrai yra tarpusavyje tinkamai suderinti.

Duomenų ir duomenų laikmenų darna:

1. Sutelkta į duomenų transakcijas. Tikrinama ar duomenys iš organizacijos gaunamų dokumentų (duomenų laikmenų) pateks, panaudojant sistemą, visiems, kuriems jų reikia ir nebus kaip nors iškreipti (pavyzdžiui, per daug apibendrinti).
2. Siekiama įsitikinti, kad organizacija gauna korektiškus duomenis, teisingai juos interpretuoja (t.y. taip pat, kaip duomenų siuntėjas) ir panaudoja juos tinkamai.

Procesų darna:

1. Tikrinama, ar kompiuterizuoti verslo procesai išlieka suderinti vienas su kitu.

Paaiškinkite, kas tai yra reikalavimų peržiūra. Kokiu tikslu ji atliekama? (4)
Reikalavimų peržiūra atliekama tam, kad peržiūrėti analizės rezultatus ir nustatyti, ar kas nors nėra praleista, ar visos problemos nustatytos ir ar pasiūlyti tinkami jų sprendimo būdai. Analizės rezultatus aprašanti dokumentacija turėtų būti parengta taip, kad ji būtų suprantama bet kuriam grupės nariui. Kitaip tariant, joje viskas turi būti išaiškinta ir joje neturėtų būti jokių dviprasmybių. Peržiūros metu tikrinama, ar iš tiesų taip yra. Prieš peržiūrą kiekvienas grupės narys privalo perskaityti dokumentaciją ir pasižymėti visas jam neaiškias vietas.

11 paskaita

Kaip yra suprantamas programų sistemos skaidymas į dalis? (1)
Sistemos hierarchijos formavimo procesas dažnai vadinamas sistemos skaidymu (partitioning) į dalis. Sistemos skaidymas į dalis yra sistemos projektavimo proceso dalis. Sistemos projektavimas yra kūrybinis procesas.

Kaip, projektuojant programų sistemą objektiškai, yra pasinaudojama koncepciniu verslo modeliu (koncepciniu dalykinės srities modeliu)? (3)
Objektinis projektavimas traktuoja verslo modelį kaip aprašą, aprašantį problemas, kurias spręsti privalo kuriamoji programų sistema (probleminės srities aprašą). Projektuojant

1. verslo modelis yra papildomas ir patikslinamas objektiniais analizės rezultatais.
2. Verslo modelis išreiškiamas konkrėčios realizavimo infrastruktūros (kompiuterinė platforma, programavimo kalba, tarpinė programinė įranga ir kt.) terminais.
3. Nagrinėjama **kokiu mastu** galima pakartotinai panaudoti turimus objektus, posistemius ir visas architektūras, kaip kurimoje PS pritaikyti tipinius projektavimo sprendimus.

Kas daroma PS koncepcinio projektavimo metu? (3)

Koncepcinis projektavimas - projektavimas aukščiausiame abstrakcijos lygmenyje. Jo metu:

- verslo modelis transformuojamas į pradinį programų sistemos modelį
- į sistemą įjungiama "dalykinė dalis";
- apibrėžiami vartotojo interfeisai, parenkami duomenų valdymo, komunikavimo ir užduočių valdymo būdai;
- parenkama koncepcinė programų sistemos architektūra;
- sprendžiama, kaip toje architektūroje įgyvendinti parinktus duomenų valdymo, komunikavimo bei užduočių valdymo būdus;
- sprendžiama, kaip pasinaudoti tipiniais projektavimo sprendimais bei turimais komponentais.

Kuo skiriasi eskizinis PS projektavimas ir jos detalusis projektavimas? (2)
Eskizinio projektavimo metu koncepciniai reikalavimai patikslinami, atsižvelgiant į konkrečią kompiuterinę platformą. Detalusis projektavimas naudoja eskizinio projektavimo metu gautą projektą ir jį išreiškia realizavimo infrastruktūros terminais:

- detalūs algoritmai ir duomenų struktūros;
- objektinių struktūrų realizavimas;
- būsenų ir jų kaitos realizavimas;
- sistemos veikimo optimizavimas;
- pakartotinas komponentų panaudojimas;
- išimtinių situacijų apdorojimas ir tt.

Papasakokite apie Coad & Yourdon architektūrą ir jos komponentus (5)
Koudo ir Jodano pasiūlyta architektūra numato, kad programų sistemą sudaro keturi posistemiai: Dalykinis posistemis, interfeiso posistemis, užduočių valdymo posistemis (UVP), duomenų valdymo posistemis(DVP). UVP ir DVP sistemoje gali ir nebūti.

<u>Koudo-Jodano architektūra:</u>

- Interfeiso posistemis bendrauja su dalykiniu posistemiu.
- Dalykinis posistemis bendrauja su duomenų valdymo posistemiu (arba tiesiai su duomenų baze)
- Dalykinis posistemis bendrauja su užduočių valdymo posistemiu (arba tiesiai su operacine sistema)

Dalykinis posistemis skirtas funkciniams reikalavimams įgyvendinti.

- DP klasės gaunamos tiesiogiai iš verslo modelio.
- Pridedama šakninė klasė, apjungianti visas dalykines klases į vieną visumą.
- Sąveikai su interfeiso posistemiu įgyvendinti, į DP pridedamas tam tikras skaičius interfeisinių klasių.

- Projektuojama sąveika su duomenų valdymo komponentu arba tiesioginė sąveika su DB. *(Tam į kai kurias klases pridedami papildomi atributai bei papildomos operacijos, kartais į DP pridedamos specialiai tam skirtos klasės.)*

Interfeiso posistemis realizuoja užduočių aprašymo kalbą ir rezultatų pateikties funkcijas.

- IP – tai tam tikra ribinės PĮ rūšis.
- Jis operuoja langais, meniu ir kitais interfeiso objektais.
- Jis transformuoja užduočių aprašus į reikalavimus suteikti atitinkamas DP teikiamas paslaugas.
- IP realizuojamas panaudojant kokią nors įvykiais valdomą architektūrą.
- Dažniausiai šį posistemį galima kurti pasinaudojant rinkoje parduodamais komponentais.

Užduočių valdymo posistemio (UVP) prisireikia tik tuomet, kuomet sistema projektuojama kaip lygiagrečiai vykdomų užduočių rinkinys (multitasking). UVP yra tarpinės programinės įrangos rūšis. Dažniausiai šį posistemį kaip komponentą galima įsigyti rinkoje.

Duomenų valdymo posistemis skirtas dalykinių klasių prieigos prie duomenų bazės interfeisui (API, application program interface) sukurti.

- DB gali būti realizuota įvairiai: kaip reliacinė DB, kaip objektinė DB, kaip failų rinkinys ir t.t.
- Jei DP realizuojamas kaip sunkiojo serverio (fat server) dalis, tarkime. PL SQL (Oracle) priemonėmis, tai DVP yra nereikalingas.
- DVP yra tarpinės PĮ rūšis
- plačiai žinomas DVP pavyzdys yra ODBC.
- DVP visuomet galima įsigyti rinkoje kaip gatavą komponentą.

Kokius PS dekompozicijos būdus jūs žinote? (1)

- Funkcinė dekompozicija
- Fizinė dekompozicija
- Dekompozicija į paslaugas
- Dekompozicija į užduotis

Paaiškinkite PS funkcinės dekompozicijos esmę.(3)

Atliekant funkcinę dekompoziciją, sistema traktuojama kaip funkcinių vienetų rinkinys. Dekompozicijos metu:

- sistema dekomponuojama į modulius,
- kiekvieną modulį dalykinėje srityje atitinka pakankamai svarbus apdorojimo žingsnis (funkcija);
- moduliai gali būti skaidomi į smulkesnius modulius.

Pvz.: lėktuvo funkcinė dekompozicija į viršutinio lygmens objektus, sistemos funkcijas ir žemo lygmens objektus:

- ***Viršutinio lygmens objektai:*** navigavimo sistema, radaro sistema, ryšio sistema, ...
- ***Sistemos funkcijos:*** rodyti trasą (radaro posistemis), palaikyti dažnį (ryšio posistemis), ...
- ***Žemo lygmens objektai:*** variklio būsena, aukštimatis, radijo švyturys, ...

Paaiškinkite PS objektinės dekompozicijos esmę. (3)

Objektinės dekompozicijos metu į sistemą žiūrima kaip į objektų rinkinį.

- Sistema dekomponuojama į klases (“objektus”).
- Dauguma klasių atitinka dalykinės srities sąvokas.
- Klasės gali būti dekomponuojamos į smulkesnes klases.

Paaiškinkite objektinio nuoseklių patikslinimų metodo esmę. (4)

Objektinis nuoseklių tikslinimų metodas (ONPM) grindžiamas abstrakcijos principu. Tai vienas iš seniausiųjų objektinių PS projektavimo metodų. Paprastu nuoseklių tikslinimų principu grindžiamas projektavimo metodas tinka projektuoti tik monolitines programas. Siekiant projektuoti objektines programas, šis metodas buvo išplėstas.

- Metodas palaiko iteratyvų programų kūrimo būdą, apjungiantį į vieną gyvavimo ciklo stadiją projektavimą ir kodavimą. Visą kūrimo laiką programa susideda iš dviejų dalių:
  - jau parašyto kodo: visa, kas iki to momento jau yra galutinai suprogramuota;
  - ketinamo parašyti kodo: visa, kas dar turi būti suprogramuota

- Interfeisas tarp šių programos dalių yra vadinamas neatliktų darbų interfeisu. Kitaip tariant, tai ketinamos programuoti programos dalies projektas. Jam priklauso esybės, kurias reikia apibrėžti artimiausioje ateityje.

ONPM išplečia tradicinį neatliktų darbų interfeisą: jam priklauso *duomenų struktūros*, *procedūros/funkcijos* ir tarp jų tekantys *duomenų srautai*. Eilinis ONPM žingsnis susideda iš tokių veiksmų:

1. iš einamojo neatliktų darbų interfeiso *išrenkamas* esybių klasteris (visos esybės susijusios su dalykinės srities objektų arba reikalinga duomenų struktūra);
2. *apibrėžiamos* tam klasteriui priklausančios esybės ir šitaip gautu apibrėžčių paketu papildoma parašytoji programos dalis;
3. *Atnaujinamas* neatliktų darbų interfeisas
4. Neatliktų darbų interfeisas *papildomas* esybėmis, reikalingomis duomenims perduoti tarp naujai atsiradusių esybių.

Paaiškinkite objektinio struktūrinio projektavimo metodo esmę. (4)
Objektinis struktūrinio projektavimo metodas (OSPM) remiasi prielaida kad turima užduočių diagrama *(use case diagram)* aprašanti programų sistemą aukščiausiame abstrakcijos lygmenyje:

- kiekvieną PS reikalavimų dokumentacijoje aprašytą užduotį turi atitikti užduotis užduočių diagramoje;
- po to, kiekviena užduočių diagramos užduotis turi būti aprašyta tos užduoties vykdymo scenarijumi ir kiekvienas toks scenarijus turi būti pavaizduotas atitinkama sekų diagrama;
- iš kiekvienos sekų diagramos generuojamos aukščiausiojo abstrakcijos lygmens klasių ir būsenų diagramos;
- visos su viena užduotimi susijusios diagramos yra grupuojamos į vieną paketą. šitaip gaunami paketai P_1, P_2, ..., P_n

Kitame žingsnyje, panaudojant standartinę užduočių modeliavimo metodiką, kiekvienas paketas yra dekomponuojamas į žemesnio abstrakcijos lygmens paketus:

- kiekvienam neelementariam sekų diagrama aprašyto scenarijaus žingsniui apibrėžiama žemesnio lygmens užduotis;
- kiekviena nauja užduotis aprašoma scenarijumi ir tas scenarijus pavaizduojamas sekų diagrama;
- kiekvienai naujai užduočiai kuriamos žemesniojo lygmens klasių ir būsenų diagramos (jos kuriamos detalizuojant aukštesniojo lygmens diagramas, kam panaudojamos tų užduočių sekų diagramos);
- kiekvienai naujai užduočiai kuriamas žemesnio lygmens paketas, grupuojantis visas tos užduoties diagramas.
- Procesas yra pabaigiamas, kada nei viename scenarijuje nebelieka nei vieno neelementaraus žingsnio. *(Žingsnis vadinamas elementariu, jei jį galima aprašyti paprastu pranešimu).*

Paaiškinkite paslaugomis grindžiamos dekompozicijos esmę (3)
Paslaugomis grindžiama dekompozicija yra naudojama paslaugas teikiančioms sistemoms, dažniausiai tai yra Web sistemos, projektuoti.

Paslaugomis grindžiama *(service-oriented)* dekompozicija susideda iš tokių žingsnių:

- ***Posistemių analizė*:** Specifikuojami tarp posistemių tekantys duomenų srautai ir kitos posistemių tarpusavio priklausomybės. Aprašomos posistemių vykdomos užduotys (use case), kurios yra traktuojamos kaip paslaugos užprašomos per posistemių interfeisus.
- ***Komponentų specifikavimas*:** Specifikuojamos paslaugas įgyvendinančių komponentų detalės: duomenys, taisyklės, naudojamos paslaugos, konfigūruojami profiliai, variacijos. Šiame žingsnyje taip pat yra specifikuojami pranešimai bei įvykiai ir apibrėžiama, kaip vyksta valdymas komponentų viduje.
- ***Paslaugų lokalizavimas*:** paslaugos susiejamos su posistemiais (lokalizuojamos posistemiuose)
- ***Paslaugų realizavimas*:** parenkama internete arba kuriama paslaugas realizuojanti programinė įranga *(Web servisai)*.

Paaiškinkite užduotimis grindžiamos dekompozicijos esmę. (3)
Užduotimis grindžiama (task-oriented) dekompozicija naudojama agentinėms sistemoms projektuoti. Ši metodika vartoja organizacijų projektavimo teorijos sąvokas: užduotis *(task)*, valdymas *(control)*, darbas *(job)*, operacija *(operation)*, vadovavimas *(management)*, koordinavimas *(coordination)* ir organizavimas *(organisation)*.

- Sistemos vykdomos užduotys yra dekomponuojamos į darbus.
- Po to darbai vėl integruojami, panaudojant darbų lokalizavimo, koordinavimo ir organizacinio pertvarkymo mechanizmus.

Kas vadinama reikalavimų lokalizavimu? (2)
Bet kuris sistemos reikalavimas turi būti lokalizuotas viename arba daugiau žemesnio lygmens komponentų. Lokalizavimo procesas užsibaigia lokalizavus visus sistemos reikalavimus.

Apibūdinkite plačiau reikalavimų lokalizavimą? Kas tai yra reikalavimų ryšio matrica?? (3)
Lokalizavus reikalavimą posistemyje, tą reikalavimą reikia suformuluoti kaip vieną ar daugiau posistemio reikalavimų.
Dažniausiai formuluojami keli reikalavimai, bet gali būti ir atvirkščiai, t.y. iš kelių posistemyje lokalizuotų reikalavimų gali būti suformuluotas vienas (išvestinis) posistemio reikalavimas.
Judant sistemos hierarchija žemyn, reikalavimai tampa vis detalesni ir konkretesni.
Atliekant reikalavimų nuleidimą žemyn, gali būti aptiktos lokalizavimo, sistemos hierarchijos formavimo ir netgi sistemos reikalavimų klaidos.

Plačiau apibūdinkite reikalavimų trasavimą? (2)
Atliekant reikalavimų nuleidimą ir lokalizavimą, reikalavimų skaičius didėja labai sparčiai. Norint įsitikinti, kad reikalavimų nuleidimo metu jokie reikalavimai neprapuolė ir neatsirado kokių nors naujų, nemotyvuotų reikalavimų, reikalavimus reikia susieti į trasas, parodant, kokie reikalavimai iš kokių buvo išvesti. Jei tenka keisti pradinius sistemos reikalavimus, pagal trasas yra nustatoma, kokius išvestinius projektavimo ir realizavimo reikalavimus tie pokyčiai palies.

Kokie interfeisai yra apibrėžiami, atliekant reikalavimų lokalizavimą ir nuleidimą žemyn? (2)
Kiekviename sistemos hierarchijos lygmenyje, atlikus reikalavimų lokalizavimą ir nuleidimą žemyn, reikia specifikuoti to lygmens elementų interfeisus. Specifikavimo procesas susideda iš dviejų žingsnių:

- Specializuojami ir detalizuojami aukštesniojo lygmens interfeisai - pvz., sistemos vartotojo interfeisai lokalizuojami posistemiuose ir nuleidžiami žemyn.
- Interfeisai papildomi elementais, reikalingais to lygmens elementų tarpusavio sąveikai

Kokius reikalavimų verifikavimo ir vertinimo metodus jūs žinote? Trumpai apibūdinkite kiekvieną iš jų. (3)
Projektavimo rezultatams *verifikuoti* ir *vertinti* yra naudojamos tos pačios procedūros kaip ir verifikuojant bei vertinant sistemos reikalavimus, t.y., peržiūra (walk-through) ir inspektavimas (inspection).

- **Peržiūra** visų pirma yra vertinimo procedūra, ji skirta kokiai nors analizuojamai medžiagai išsiaiškinti ir perprasti. Todėl peržiūros grupės dalyviai yra skatinami užduoti kuo daugiau klausimų asmenims, pristatinėjantiems tą medžiagą. Prieš grupės susitikimą peržiūrima medžiaga yra išdalinama visiems peržiūros grupės nariams. Peržiūros metu, pristatantysis trumpai apžvelgia pristatomus darbo rezultatus, po to grupės nariai, atsakingi už atskirų klausimų referavimą, žingsnis po žingsnio apžvelgia jo peržiūrėtą medžiagą. Labai svarbu, kad peržiūros metu vyrautų laisva, neformali atmosfera.
- **Inspektavimas** yra verifikavimo procedūra, ja siekiama atrasti darbo metu padarytas klaidas ir užtikrinti aukštą rezultatų kokybę. Inspektavimo grupę sudaro lygiateisiai nariai, gerai išmanantys darbo su reikalavimais procedūras ir susipažinę su projekte naudojamais standartais. Inspektuojama medžiaga (reikalavimai, projektinė dokumentacija, programų tekstai ir kt.) išdalijama grupės nariams keletą dienų prieš grupės posėdį. Kiekvienas grupės narys gauna klausimyną, į kurio klausimus jam reikia atsakyti. Klausimyno pobūdis priklauso nuo to, už ką atsakingas yra tas grupės narys. Posėdžio metu, kiekvienas narys pristato savo išvadas. Jas apibendrinus, gaunamas bendras inspektuojamos medžiagos įvertinimas.

Pagal kokias charakteristikas vertinama projektavimo kokybė programų sistemos prižiūrimumo požiūriu? (1)

Projektavimo kokybė vertinama pagal:

- ***Komponentų rišlumą.*** Rišlumas kaip projektavimo kokybės vertinimo kriterijus suformuluotas nepakankamai griežtai.
- ***Komponentų sankibą.*** Silpnesnė sankiba reiškia geresnę projektavimo kokybę.
- ***Projekto suprantamumą.*** Jis priklauso nuo komponentų rišlumo, pavadinimų ir dokumentacijos išsamumo.
- ***Projekto adaptuojamumą.*** Projektas nesunkiai adaptuojamas, jei komponentai yra silpnai sukibę.

Kokius rišlumo lygmenis jūs žinote? (4)

Rišlumo lygmenys:

- ***Atsitiktinis rišlumas*** (silpnas) - komponento dalys niekaip nesusietos tarpusavyje.
- ***Loginis rišlumas*** (silpnas) - komponentas realizuoja kelias pagal panašius algoritmus veikiančias funkcijas.
- ***Rišlumas pagal laiko momentą*** (silpnas) - visos komponento dalys vykdomos tuo pat metu, bet jos naudojamos skirtingiems tikslams.
- Procedūrinis rišlumas (silpnas) - komponento dalys nuosekliai vykdomos viena po kitos (sujungtos valdymo grandine).
- ***Komunikacinis rišlumas*** (vidutinis) - visos komponento dalys arba apdoroja tą pačią įeigą, arba kuria tą pačią išeigą.
- ***Nuoseklus rišlumas*** (vidutinis) - komponento dalys siejamos duomenų grandine (vienos dalies išeiga yra įeiga kitai daliai).
- ***Funkcinis rišlumas*** (stiprus) - visos komponento dalys skirtos tai pačiai funkcijai realizuoti.
- ***Objektinis rišlumas*** (stiprus) - komponento dalys – tai operacijos, vykdomos ant tos pačios duomenų struktūros (komponentas realizuotas kaip objektas).

Kaip kinta komponentų (modulių) sankiba? Kaip siejasi tarpusavyje sankiba ir paveldėjimas? (3)

Komponentų sankiba išauga, jeigu jie naudojasi tomis pačiomis duomenų struktūromis arba jeigu jie mainosi valdančiąja informacija. Sankiba mažinama decentralizuojant sistemos būseną, pavyzdžiui, išskirstant ją po objektus ir keičiantis informacija tik per gerai apibrėžtus interfeisus, geriausiai, siunčiant pranešimus.

Objektinėse sistemose **sankiba** esti nedidelė, nes sistemos būsena yra išskirstyta po objektus ir objektai komunikuoja keisdamiesi pranešimais. Tačiau, objektinėse sistemose klasės yra sukibusios su savo poklasiais (**paveldėjimas**). Keičiant klasės atributus ar jos operacijas, tenka daryti pakeitimus ir visose tos klasės poklasiuose. Visa tai sužiūrėti yra pakankamai sudėtinga.

Paaiškinkite, kaip yra suprantamas terminas "projekto suprantamumas? (2)

Projektinės dokumentacijos suprantamumas priklauso nuo keleto dalykų:

- ***Komponentų rišlumo.*** Kiek paprasta perprasti kiekvieną iš komponentų?
- ***Komponentų pavadinimų.*** Ar jie yra prasmingi?
- ***Dokumentacijos išsamumo.*** Ar projektavimo sprendimai pakankamai gerai dokumentuoti?
- ***Sudėtingumo.*** Kiek sudėtingi algoritmai, kurie įgyvendinami sistemoje?

Esant dideliam sistemos sudėtingumui, sunku perprasti sistemos komponentų sąryšius.

Paaiškinkite, kaip yra suprantamas terminas "projekto adaptuojamumas". Kaip siejasi tarpusavyje adaptuojamumas ir paveldėjimas? (3)

Adaptuojant projektą, jis yra keičiamas, todėl reikia išanalizuoti kiekvieno pakeitimo pasekmes. Čia labai padeda trasavimo matricos. Projektas yra nesunkiai adaptuojamas, jeigu:

- sistemos komponentai yra silpnai sukibę;
- projektavimo rezultatai yra išsamiai dokumentuoti ir dokumentacija nėra pasenusi;
- projektinės dokumentacijos lygmenys surišti tarpusavyje (trasos);
- visi komponentai yra pakankamai rišlūs.

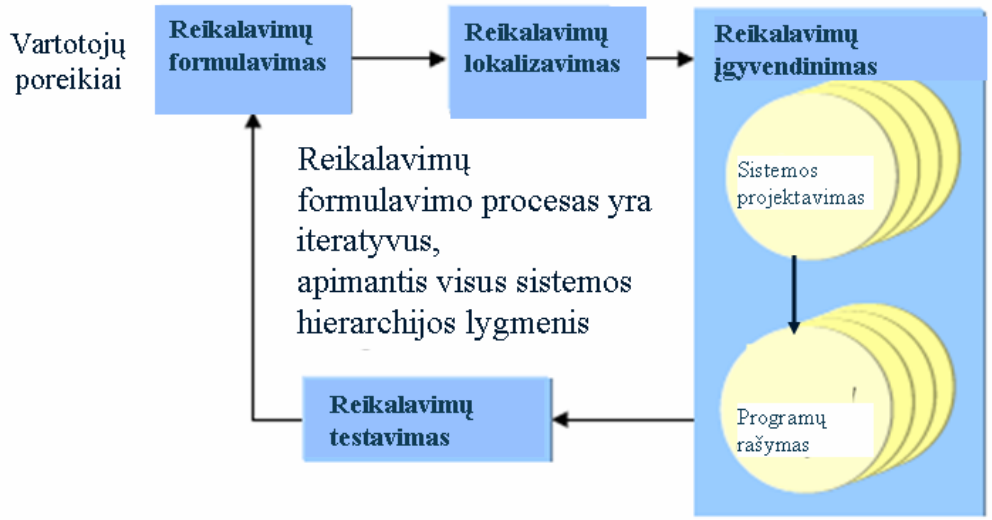
Paveldėjimas iš esmės pagerina **adaptuojamumą**. Komponentą galima adaptuoti jo nekeičiant, bet vietoj to sukuriant atitinkamą poklasį, kuriame ir padaromi reikiami pakeitimai. Tačiau, didėjant paveldėjimo hierarchijos lygmenų skaičiui, auga ir sistemos sudėtingumas. Prisireikia ją periodiškai peržiūrėti ir pertvarkyti.

Paaiškinkite, kuo skiriasi testavimas ir derinimas. (3)
Derinimas yra kodo testavimas, klaidų radimas, jų analizė ir taisymas. Tai yra procesas, kuris tesiamas, iki kol klaidų neberandama. Testavimas yra modulių, agregatų, posistemių testavimas, tikrinant, ar jie tenkina jiems suformuluotus reikalavimus, sistemos testavimas - tikrinant funkcines ir nefunkcines sistemos savybes. Modulių ir agregatų testavimą atlieka programuotojai, posistemių ir sistemos testavimas atliekamas integruojant sistemą. Taigi testavimas yra sistemos ir jos dalių tikrinimas pagal esamus reikalavimus, tuo tarpu derinimas yra klaidų kode paieška ir taisymas.

Kokias testavimo stadijas jūs žinote? Trumpai apibūdinkite kiekvieną iš jų. (3)

- Testavimo žingsniai:
  1. ***Modulių testavimas:*** tikrinama ar visi baziniai sistemos moduliai tenkina jiems suformuluotus reikalavimus..
  2. ***Agregatų testavimas:*** tikrinama ar sistemos agregatai (tampriai susijusių modulių grupės) tenkina jiems suformuluotus reikalavimus.
  3. ***Posistemių testavimas:*** tikrinama ar sistemos posistemiai tenkina jiems suformuluotus reikalavimus.
     1. Pagrindinis dėmesys skiriamas interfeisų testavimui.
  4. ***Sistemos testavimas:*** tikrinamos funkcinės ir nefunkcinės sistemos savybės
     1. atliekamas vykdytojo kaip vidiniai sistemos bandymai..
  5. ***Baigiamasis*** *(acceptance)* ***testavimas:*** atliekamas perduodant sistemą užsakovui *(baigiamųjų bandymų metu)* pagal su užsakovu suderintą testavimo planą.

Paaiškinkite, kaip yra susieti tarpusavyje programų sistemos reikalavimai ir jos testavimas. (3)
Reikalavimai ir testavimas:



Aprašymas

Paaiškinkite, ką reiškia terminas "padengimas testais". (3)
PS testavimas suprantamas kaip sistemos tikrinimas, ar sukurta sistema tenkina reikalavimus. Testavimą sudaro testų rinkinys, kuris padengia kuriamos PS reikalavimus. Testas gali padengti (t.y., patikrinti testuojant) vieną ir daugiau reikalavimų. Reikalavimai gali būti tikrinami vieno arba daugiau testų. Jeigu sistemoje lieka nors vienas nepatikrintas reikalavimas, sakoma, kad sistema nėra pilnai padengta testais.

Kaip suprantamas terminas "programų sistemos gyvavimo ciklo modelis" (PSGCM)? Kokias veiklas paprastai apima PSGCM? (5)
Kuomet buvo pradėti kurti pirmieji programų sistemų inžinerijos metodai, buvo suvokta programų sistemų kūrimo proceso apimtys ir sudėtingumas. Tapo aišku, kad reikia susisteminti sistemų kūrimo procesą, t.y. kad reikalingos sistemų kūrimo metodikos. Tos metodikos buvo pavadintos PS gyvavimo ciklo (PSGC) modeliais. PS kūrimo procesas paprastai apima tokias veiklas:

1. Kompiuterizuojamo verslo sistema yra analizuojama ir atskleidžiami jos trūkumai
2. Formuluojami kuriamos sistemos reikalavimai
   1. Jie turi būti tokie, kad sistema padėtų pašalinti nustatytus verslo sistemos trūkumus.
3. Programų sistemos projektavimas

1. Be kita ko, numatoma su koki technine įranga ir su kokia operacine sistema dirbs kuriamoji programų sistema, kaip ji veiks kompiuterių tinkluose, kokiomis programavimo kalbomis bus programuojama ir kaip bus apsaugota nuo neteisėto naudojimo.

4. Programų sistemos sukūrimas ir jos diegimas.
    1. Sukuriami ir instaliuojami visi sistemos komponentai.
    2. Apmokomi sistemos vartotojai, realiomis darbo sąlygomis vertinami sistemos patikimumas ir našumas.
    3. Jei reikia, atliekami sistemos pataisymai.
5. Programų sistemos perdavimas eksploatacijai.
    1. Tai gali būti atlikta įvairiais būdais
        1. Palaipsniui, viena po kitos, rankinės darbo procedūros keičiamos kompiuterizuotomis.
        2. Jei sistema keičia seną programų sistemą, jai palaipsniui perduodamos tos sistemos funkcijos.
        3. Kartais gali būti pigiau "vienu ypu" seną sistemą pakeisti nauja.
6. Pradėjus sistemą eksploatuoti, ją reikia nuolat stebėti bei vertinti ir galbūt papildyti ar keisti.Tai vadinama programų sistemos priežiūra (maintenance)
    1. Visi sistemos vartotojai turi būti laiku informuoti apie visus padarytus pakeitimus.
    2. Programų sistemos aptarnavimas ir jos priežiūra yra visiškai skirtingi dalykai.

Kokius programų sistemos gyvavimo ciklo modelius jūs žinote? (1)

Gyvavavimo modeliai:

1. "Koduok ir atiduok" modelis
2. "Krioklio" modelis
3. Prototipus naudojantis modelis
4. Evoliucinis modelis
5. Greito kūrimo (rapid application development (RAD)) modelis
6. Grupinio kūrimo (joint application development (JAD)) modelis
7. "Sinchronizuok ir Stabilizuok" (synchronise-and-stabilise) modelis
8. Spiralinis modelis
9. Formalaus PS kūrimo modelis
10. Komponentinis modelis (reuse-oriented)

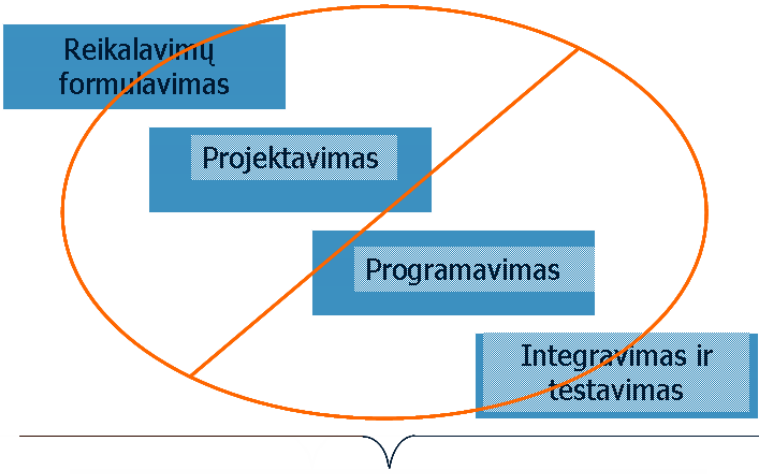
Trumpai apibūdinkite programų sistemos gyvavimo ciklo modelio "Koduok ir atiduok" esmę. (1)

Modelis "Koduok ir atiduok" (Code & Fix model) yra istoriškai susiklostęs pats pirmasis programuotojų darbo būdas. Juo vadovaujantis, rašomas kodas ir atiduodamas užsakovui. Nėra jokios analizės ir jokio projektavimo. Tokio modelio problemos:

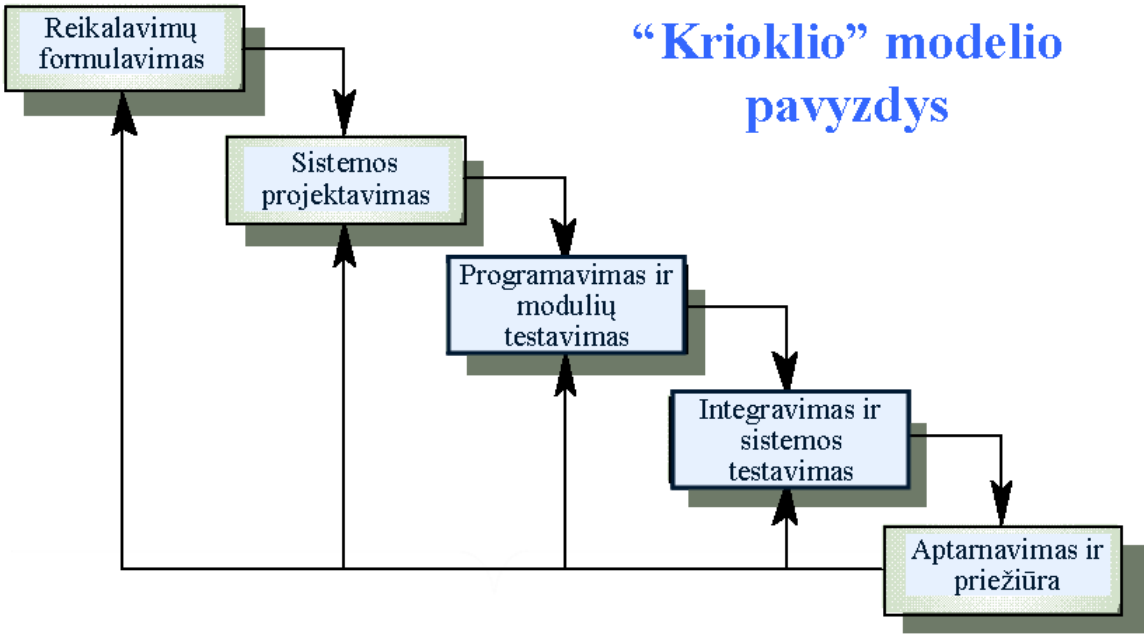
- kodas netinkamai struktūrizuotas
- sistema prastai tenkina operacinius vartotojų poreikius (neatlikta dalykinės srities analizė)

Apibūdinkite "krioklio" modelio esmę. (5)

"Krioklio" modelis (waterfall model) dar vadinamas klasikiniu PSGCM. Grindžiamas "iš viršaus žemyn" PS kūrimo paradigma. Proceso stadijos (jų gali būti skirtingas skaičius) vykdomos nuosekliai viena po kitos. Modelis pasiskolintas iš automobilių pramonės. Šiuo metu plačiai naudojamas ir karinėje bei kosmoso industrijoje. Geras vadybiniu požiūriu, nes nėra nei persidengiančių, nei lygiagrečiai vykdomų stadijų.



Tipinės "krioklio" modelio stadijos



"Krioklio" modelio pavyzdys

Kaip programų sistemos gyvavimo ciklo modeliuose yra panaudojami prototipai? Kas vadinama prototipu? (3)

Prototipas yra pradinė programų sistemos versija, demonstruojanti kokius nors jos dalykinius *(dažniausiai, interfeiso)* ar projektinius ypatumus. Prototipus naudojantis gyvavimo ciklo modelis numato prototipų *(grubių būsimos sistemos aproksimacijų)* konstravimą, testavimą ir perdirbimą, žingsnis po žingsnio artėjant prie galutinės sistemos versijos. Modelis grindžiamas iteracine programų sistemų kūrimo paradigma.

Kaip programų sistemos gyvavimo ciklo modeliuose yra panaudojami išmetamieji prototipai (maketai)?(3)

Maketavimas (išmetamų prototipų naudojimas): maketai kuriami, siekiant išsiaiškinti sistemos reikalavimus.

Pradedama nuo menkai perprastų reikalavimų.
Konstruojamas tuos reikalavimus demonstruojantis maketas.
Užsakovas pareiškia savo pastabas.
Maketas patikslinamas.

Tai kartojasi iki tol, kol užsakovas nebeturi pastabų. Maketų naudojimas tinkamas tada, kai:

sunku suformuluoti tikslius reikalavimus (dažniausiai, interfeiso).
turimos geros maketų konstravimo priemonės.

Apibūdinkite tiriamojo ir evoliucinio maketavimo esmę. (5)

Tiriamasis *(exploratory)* ***maketavimas:*** Dirbama kartu su būsimaisiais vartotojais. Maketas imituoja išorinę sistemos elgseną, modeliuoja visų jo funkcijų vykdymą, bet nei viena iš funkcijų iš tiesų nėra realizuota. Maketo bandymai atliekami kartu su vartotojais, bandymų rezultatai analizuojami irgi kartu. Tikslas – aprobuoti funkcinius sistemos reikalavimus ir vartotojo interfeiso reikalavimus. Maketas keičiamas tol, kol

vartotojai pripažins, kad jis veikia taip, kaip turėtų veikti būsimoji programų sistema. Dirbdami su maketu, vartotojai pamažu suvokia, kokios priemonės jiems iš tiesų reikia jų kasdieniniam darbui patobulinti. Taip žingsnis po žingsnio tikslinami kuriamosios sistemos reikalavimai. Pradedama nuo geriausiai perprastų ir mažiausiai ginčų sukeliančių reikalavimų. Į kiekvieną naują maketo versiją pridedama tai, kas užsakovo nuomone yra labiausiai reikalinga.

Evoliucionuojantis maketavimas *(evoliucionary development)*: Šiuo atveju maketas nėra išmetamas, bet traktuojamas kaip būsimosios sistemos prototipas. Maketai (tiksliau, funkcijų realizacija) tikslinami žingsnis po žingsnio ir šitaip artėjama prie galutinio sistemos varianto. Kaip ir tiriamojo maketavimo atveju, kiekvienas naujas maketas vis labiau artėja prie galutinio varianto ne tik realizavimo išbaigtumo, bet ir funkciniu požiūriu. Naudojant evoliucinio maketavimo metodiką, galima tikslinti ir nefunkcinius (našumo, patikimumo ir kt.) sistemos reikalavimus. Problemos: Neskaidrus procesas (stadijos persipina); Dažnai sistema gaunasi prastai struktūrizuota; Programuotojai turi turėti specialius įgūdžius (pvz., mokėti naudotis prototipų kūrimo priemonėmis). Tinkamumas: Mažoms ir vidutinio dydžio interaktyvioms sistemoms; Didelių sistemų komponentams (pvz., vartotojo interfeisui); Trumpai gyvuojančioms sistemoms.

Trumpai apibūdinkite greito PS kūrimo modelio esmę. (1)
(rapid application development (RAD)) Modelis daro prielaidą, kad kūrimo procesą galima pagreitinti: panaudojant fokuso grupes; panaudojant prototipus ir tipinius projektavimo sprendimus; griežtai prisilaikant suplanuotų terminų; mažiau formalizuojant darbo procedūras.

Trumpai apibūdinkite grupinio PS kūrimo modelio esmę. (1)
Šis modelis numato užsakovų ir vartotojų dalyvavimą projektavimo ir kituose sistemos kūrimo procesuose. Tai padaroma sukuriant bendras darbo grupes ir vykdant projektavimą tų grupių posėdžiuose (JAD - joint application development - sessions).

Trumpai apibūdinkite programų sistemos gyvavimo ciklo modelio "Sinchronizuok ir stabilizuok" esmę. (1)
Šis modelis numato, kad savarankiškos grupės lygiagrečiai kuria skirtingus modulius, dažnai derindami tarpusavyje tų modulių kodus ir nuspręsdami įšaldyti (stabilizuoti) tam tikrus projektavimo sprendimus.

Apibūdinkite evoliucinio programų sistemų gyvavimo ciklo modelio esmę. Kas vadinama ekstremaliuoju programavimu? (4)
Dirbant pagal šią metodiką, užsakovui sistema pateikiama ne iš karto, bet pateikiant jos branduolį ir prieaugius *(increments)*. Privalumai:

Su kiekvienu prieaugiu sukuriama ir pateikiama užsakovui papildoma vertė, vartotojai gali pradėti dirbti su (funkciniu požiūriu neišsamia) sistema daug anksčiau, negu naudojant kitus PSGCM.

Pradiniai prieaugiai gali būti panaudoti kaip maketai ir padėti tikslinti sistemos reikalavimus.

Sumažėja projekto žlugimo rizika.

Vartotojų požiūriu svarbiausios paslaugos yra realizuojamos pradiniuose prieaugiuose, kurie yra testuojami išsamiausiai (pakartotinai testuojami pridėjus kiekvieną naują prieaugį).

Ekstremalusis programavimas: Tai nauja ir šiuo metu labai populiari evoliucinio modelio panaudojimo metodika. Jos esmė – kiek įmanoma mažesnės apimties prieaugiai. Galima sakyti, kad tai yra kodo nuolatinio tobulinimo metodika. Kiti ypatumai: aktyvus vartotojų dalyvavimas sistemos projektavimo darbuose ir programavimas poromis (pair-wise programming).

Apibūdinkite spiralinio programų sistemų gyvavimo ciklo modelio esmę. (5)
Šis modelis kombinuoja "krioklio" modelio ir prototipų panaudojimo metodikas. Spiralinis modelis rekomenduotinas dideliems, brangiems ir sudėtingiems projektams. Šiuo atveju procesas yra spiralinio, o ne nuoseklaus (viena po kitos vykdomos veiklos) pobūdžio. Spiralės vingiai atitinka proceso stadijas. Modelis neturi fiksuotų stadijų, jos parenkamos kiekvienam projektui, priklausomai nuo jo poreikių. Į modelį išreikštiniu būdu yra įvesti rizikos analizės ir eliminavimo veiksmai. Spiralinio modelio sektoriai:

<u>Tikslų, alternatyvų ir ribojimų nustatymas:</u>

Definuojami einamosios stadijos tikslai, jų įgyvendinimo alternatyvos ir ribojimai.

<u>Rizikos vertinimas ir eliminavimas:</u>

Analizuojami rizikos veiksniai ir numatomos priemonės rizikai sumažinti.

<u>Kūrimas ir vertinimas:</u>

Gali būti parinktas bet kuris universalus PSGCM.

Planavimas:

Atliekama stadijos rezultatų peržiūra ir planuojama kita stadija (t.y. naujas spiralės vingis).

Apibūdinkite formalaus programų sistemų kūrimo modelio esmę(5)

Grindžiamas sintezės paradigma t.y. pažingsnis matematinės sistemos specifikacijos transformavimas į veikiančią sistemą. Transformacijos yra tokios, kad jos išsaugo teisingumą, dėl ko atpuola būtinybė tikrinti ar sistema tenkina reikalavimų specifikaciją. Problemos: Reikalingi aukštos kvalifikacijos specialistai, turintys specialius įgūdžius. Kai kuriuos sistemos aspektus, pavyzdžiui, vartotojo interfeiso reikalavimus, sunku formaliai specifikuoti. Panaudojamumas: Kritinės sistemos, ypač tokios, kurios kelia grėsmę žmonių gyvybei. Žingsniai: Reikalavimų specifikavimas; Specifikacijos formalizavimas; Formalios transformacijos; Sistemos integravimas ir testavimas.

Apibūdinkite komponentinio PS kūrimo modelio esmę. (4)

Grindžiamas komponentine PS kūrimo paradigma. Sistema renkama iš gatavų komponentų, paprastai, įsigyjamų rinkoje (COTS: Commercial-off-the-shelf). Proceso stadijos: komponentų analizė; reikalavimų modifikavimas; sistemos projektavimas komponentų pagrindu; komponentų integravimas.

Kokias programų sistemų gyvavimo ciklo modelio reikalavimų kategorijas jūs žinote. Trumpai apibūdinkite kiekvieną kategoriją. (4)

Panaudojimo srities reikalavimai nusako kokio tipo sistemoms bei projektams pritaikytas PSGCM.

Techniniai ir vadybiniai reikalavimai nusako PSGCM numatomų užduočių tipus ir proceso kuriamų rezultatų (deliverables) tipus.

Panaudojamumo reikalavimai nusako, kokiais būdais projekto vykdytojai gali pasinaudoti PSGCM ir ką reikia padaryti, kad tuo modeliu būtų paprasčiau pasinaudoti.

Diegimo reikalavimai nusako, ką reikia padaryti, diegiant PSGCM konkrečiame projekte.