Programų sistemų Inžinerija I

Prof. A. Čaplinsko 2009m. dėstomo kurso,
„Programų sistemų inžinerija I“ paskaitų konspektas

Parengė Kęstutis Matuliauskas

2010 m. sausio 14 d.

Užrašams

Pratarmė

A. Čaplinsko skaitomam kursui – „*Programų sistemų inžinerija I*“, 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

Trumpiniai

ACM – Association for Computing Machinery. Įkurta 1947. Yra pati svarbiausia IT organizacija pasaulyje, siekianti lavinti IT srityje profesionalus ir studentus.

IEEE – Institute of Electrical and Electronics Engineers. Ne pelno siekianti organizacija, pirmaujanti pasaulyje technologijų pažangos asociacija.

ISO – International Organization for Standartization. Pirmaujanti pasaulyje tarptautinių standartų kūrėja.

SEI – Software Engineering Institute. PSI institutas – tai JAV vyriausybės(gynybos departamento) finansuojamas mokslo ir technologinių tyrimų centras. Svarbiausia PSI instituticija pasaulyje.

ESM – kompiuteris. Kitaip dar vadinamas „**E**lektronine **S**kaičiavimo **M**ašina“.

PC/AK/PK – Asmeninis kompiuteris(„Personal Computer“).

GNU – GNU's Not Unix.

OMT – objektinio modeliavimo metodika.

RUP – Rational Unified Process — kartotinio programinės įrangos kūrimo metodika, sukurta įmonės Rational Software, nuo 2003 m. priklausančios IBM.

UML – Unified Modeling Language —Vieninga modeliavimo kalba – modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.

OOP/OP – Objektinis programavimas(*„Object-Oriented Programming“*)

PĮ – Programinė įranga.

TĮ – Techninė įranga

TeX - yra teksto formatavimo kalba bei įrankiai. Šiandien tai populiariausia sistema moksliniams tekstams*, fizikos, matematikos srityse,* rengti. Su **Latex** paketu, TeX dažnai naudojamas kaip maketavimo priemonė.

SI – Sistemų inžinerija.

IS – Informacinė sistema.

ISI – Informacinių sistemų inžinerija.

VPR – Verslo proceso reinžinerija.

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

PE/PA – Programavimo aplinka*(„Programming enviroment“)*

IDE – PĮ integruota kūrimo aplinka*(„integrated development environment“)*

SDE – PĮ kūrimo aplinka*(„software development environment“)*

SDK – PĮ kūrimo įrankių rinkinys, skirtas supaprasti darbą programuotojui*(„software development kit“)*

IPSE/IPPA – Integruotosios projekto palaikymo aplinkos*(„Integrated Project Support Environment“)*

CASE/KPSI – Kompiuterizuotos PS inžinerijos*(„Computer-Aided Software Engineering“)*

VM – Virtuali mašina; **CPU** – centrinis sistemos procesorius.

RI – Reikalavimų inžinerija

SNS – Sistemos naudojimo scenarijus

VPS – Verslo poreikių specifikacija

DT – Duomenų tipas; **DB** – Duomenų bazė;

ADT – Abstraktus duomenų tipas, **ADT savybės** – abstrakčios duomenų tipo savybės

SWOT – Stripriosios/silpnosios pusės, galimybės ir grėsmės*(„Strengts-Weaknesses-Oportunities-Threads“)*

Apdovanomai

5. Kokius apdovanojimus, teikiamus už mokslinius pasiekimus programų sistemų inžinerijos srityje, jūs žinote? (1)

ACM SIGSOFT Software Engineering Awards: Distinguished Service Award ir Outstanding Research Award; IEEE Harlan D. Mills Award - už teorijos taikymą praktikoje;
IEEE Software Process Achievement Award - už reikšmingą PS kūrimo patobulinimą savo įmonėje

6. Koks apdovanojimas yra vadinamas „Nobelio premija informatikams“? Kam šis apdovanojimas buvo suteiktas už pasiekimus programų sistemų inžinerijos srityje? (1)

Turing Award. Kasmet ACM skiriamas apdovanojimas asmeniui už techninio pobūdžio pasiekimus svarbius visai informatikų bendrijai. $250.000 prizą skiria Intel ir Google.

- *1978 Robert Floyd*: už efektyvios ir patikimos programinės įrangos kūrimo metodikas;
- 1999 *Frederick Brooks*: už kompiuterių architektūras, operacines sistemas, programų sistemų inžineriją.

7. Kokiu apdovanojimu apdovanota Java? Kada? (1)

ACM Software System Award: 2002 Java

Standartai

12. Kokius programų sistemų inžinerijos klausimams skirtus dokumentus yra paskelbusi IEEE? (1)

Standard Glossary of Software Engineering Terminology. (PSI terminijos standartas - žodynas)

Recommended Practice for Software Requirements Specifications. (Programinės įrangos reikalavimų specifikacijos rekomendacinis aprašymas)

Guide for Developing System Requirements Secifications. (Reikalavimų specifikacijos formavimo vadovas)

System Definition - Concept of Operation Document. (Sistemos veikimo koncepciją išdėstančio dokumento turinio ir formato aprašas)

Standard for Application and Management of System Engineering Process. (Sistemų inžinierijos proceso aprašas)

13. Kokius programų sistemų inžinerijos standartus yra parengusi ISO? (1)

ISO/IEC Standard 12207: Information Technology, Software Life Cycle Processes.
ISO/IEC Standard 9126: Software Engineering - Product quality
–*Part 1:* Quality model
–*Part 2:* External metrics
–*Part 3:* Internal metrics
–*Part 4:* Quality in use metrics.
ISO/IEC 14598: Information Technology - Evaluation of Software Products
ISO/IEC 18019: Guidelines for the design and preparation of software user documentation
ISO/IEC 15910: Software user documentation process
ISO 13407: Human-centred design processes for interactive systems
ISO TR 16982: Usability methods supporting human centred design
ISO TR 18529: Ergonomics of human-system interaction - Human-centred lifecycle process descriptions
ISO/IEC 14102: Guideline for the Evaluation and Selection of CASE Tools

Žurnalai

8. Kokiame žurnale skelbiama oficiali ACM informacija? (1)

Communications

9. Kokį studentams skirtą e-žurnalą leidžia ACM? Apie ką jame rašoma? (1)

ACM Crossroads. Kompiuterių mokslo temos studentams.

10. Koks ACM leidžiamas mokslinis žurnalas skirtas programų sistemų inžinerijos klausimams? (1)

Transactions on Software Engineering and Methodology

11. Koks IEEE leidžiamas mokslinis žurnalas skirtas programų sistemų inžinerijos klausimams? (1)

Transactions on Software Engineering

Istorija

15. Kurie metai laikomi PSI, kaip savarankiškos disciplinos, pradžia? Kodėl? (2)

1968 m.. Jais įvyko NATO konferencija programų sistemų inžinerijos klausimais.

Nuo 1985 iki 1995 metų programų sistemų inžinerija tapo savarankiška disciplina, besiskiriančia tiek nuo teorinės informatikos, tiek ir nuo tradicinių inžinerijos šakų

14. Kas toks parašė laišką „Go To Statement Considered Harmful"? Kokiais metais jį išspausdino „Communication of the ACM"? Kuo šis laiškas yra svarbus? (2)

Edsger Dijkstra. Laiške pareiškė, kad vyksta programavimo krizė, o ją sukėlė netvarkingas operatoriaus *Go To* naudojimas. Laiškas buvo vienu iš svarbiausių argumentų surengti 1968 m. NATO mokslinio komiteto finansuojamą konferenciją, kuri laikoma programų sistemų inžinerijos, kaip savarankiškos disciplinos, pradžia.

16. Kokia knyga pradėjo PSI erą? (1)

„***Structured Programming***"(1972), O. Dahl, E. Dijkstra, C. Hoare. – pirmoji knyga PSI klausimais.

Pap 1. „No Silver Bullet" straipsnis. Kas ir kada parašė? Apie ką buvo rašoma (2)

1987 m. straipsnį parašė Fred Brooks. Jame teigiama, kad jokia individuali programavimo technologija ar joks praktinis metodas, per 10 metų, nepadidins programuotojų darbo našumo 10 kartų, bei aiškinama, kodėl taip yra.

Pap 2.(4 paskaita). Kas pirmasis pasiūlė didinti darbo našumą specializuojant dirbančiuosius? Kada? (2)

XVIII amžiuje amerikiečių ekonomistas Adam Smith pasiūlė esminę įdėją, kaip padaryti grupinį darbą efektyvesniu. Ši įdėja – darbo skaidymas į smulkias užduotis ir vykdytojų specializacija pagal užduočių pobūdį.

Pap 3.(4 paskaita). Kas pirmasis pasiūlė naudoti gamyboje konvejerius? Kada? (1)

Henry Ford, 1914 m.. Jo automobilių gamykla yra vadovėlinis perėjimo prie konvejerinių darbo metodų pavyzdys. Tobulino Taylor'o metodus.

Terminai

Artefaktas – darbo produktas (susiję su RUP) - IBM pakeitė sąvoką „artefaktas“ į sąvoką „darbo produktas“.

Menas – tai veikla išsiskirianti 2 požymiais: kūrybinis veiklos pobūdis ir estetinė kuriamojo objekto vertė.

Amatas – tai tam tikrų praktinių įgūdžių reikalaujanti veikla, gebėjimą gerai daryti kokį nors produktą.

Mokslas – tai žinios arba žinių sistema apie bendrąsias tiesas ar bendruosius dėsnius, paprastai gautus ir patikrintus moksliniu metodu. Mokslas “maitina" inžineriją ir, mažesniu mastu, amatą bei meną.

Inžinerija – tai:

- mokslas, praktiškai pritaikantis turimas žinias, įvairiose praktinėse sferose - įrengimų, mašinų ir kitokių įtaisų, technologinių procesų projektavime, kūrime
- mokslo, formalių metodų, įrankių ir turimos patirties panaudojimas suprojektuotiems objektams kurti (gaminti).
- mokslo žinių ir klaidų ir bandymų metodo panaudojimas sistemoms projektuoti.

Reinžinerija – egzistuojančios sistemos analizės ir modifikavimo procesas, kuriuo siekiama iš esmės pertvarkyti tą sistemą.

Atvirkščioji inžinerija – egzistuojančios sistemos analizės procesas, kuriuo siekiama atkurti jos projektinę dokumentaciją.

Modelis – konkrečią tikslinę paskirtį turintis supaprastintas kokio nors originalo (realaus pasaulio sistemos) analogas, atspindintis to originalo struktūrą ir/arba elgseną.

Įmonė – viena ar daugiau organizacijų, jungiamų bendra paskirtimi, siekiais ar tikslais, bei kuriančių kokią nors bendrą išeigą, pvz. produktą ar paslaugas.

Darbų srautas – tai įmonėje vykdomų tam tikrų užduočių seka, skirta kokiai nors įmonės išeigai kurti.

Įmonės sistema - tai tarpusavyje susijusių verslo sistemų, *palaikomų atitinkamų informacinių ir programų sistemų*, rinkinys, apimants bent vieną verslo sistemą, ją palaikančias IS ir jų PĮ.

Funkcinis vienetas – agentas, gebantis vykdyti kokią nors operaciją (atlikti darbą). Agentas gali būti - asmuo, veikiantis tam tikrame vaidmenyje, padalinys, įrenginys, programų sistema ir pan.

Organizacija – funkcinių vienetų rinkinys.

Organizacinis procesas – procesas, skirtas kurti ir palaikyti organizacines struktūras.

Sistema – tai kartu veikiančių dalių rinkinys, kuris, žiūrint iš pašalies, atrodo kaip viena visuma ir turi aiškias ribas. *Pvz: komp. sistema, informacinė sistema, programų sistema, pastatas, automobilis.*

Sistemos reikalavimai – tai sutartimi arba kokiu nors kitu juridinę galią turinčiu dokumentu numatytos kuriamos sistemos savybės, tačiau reikalavimais nenurodoma kokius konkrečius sprendimus naudoti kuriant.

Sistemos aplinka – tai išorėje esančių daiktų, veikiančių sistemą ar jos veikiamų, visuma.

Sistemos kontekstas – tie sistemos aplinkos aspektai, nuo kurių, kaip manoma, priklauso sistemos sėkmė.

Sistemos interfeisas – tai užsakovą dominančios funkcinės ir fizinės savybės, kurių negalime priskirti nei sistemai, nei jos aplinkai.

Sistemos architektūra – tai toks sistemos komponentų organizavimo būdas, kuris įgyvendina sistemos reikalavimus ir užtikrina jos sąveiką su aplinka.

Architektūrinis sistemos projektas – tai koncepcinio lygmens struktūrinis kuriamos sistemos modelis, sudarytas tam, kad juo vadovaujantis sistema būtų konstruojama.

Elgsenos projektas – tai aprašas, kaip turėtų veikti sistema vykdydama reikalavimus, ja pasinaudoti norinčio vartotojo (*jūzerio*) terminais, neliečiant vidinių realizavimo detalių.

Dalykinė sritis*(sistemos)* – tai sistemos panaudos sritis. *Pvz.: laikrodžiui dal. sritis – "matavimo prietaisai".*

Probleminė sritis*(sistemos)* – tai tikslinė sistemos paskirtis. *Pvz.: laikrodžiui – "tikslaus laiko rodymas".*

Sistemų inžinerija – tai tarpdisciplininė inžinerijos mokslų šaka, kurioje visi sistemos kūrimo klausimai nagrinėjami labai bendru lygmeniu, atsiribojant nuo konkretaus sistemos pobūdžio.

SI*(kaip technologinis procesas*) – tai technologinis procesas, skirtas transformuoti operacinius vartotojo poreikius ir/arba sistemos reikalavimus į veikiančią sistemą, apimant veiklas, susijusias su sistemos projektavimu ir projekto įgyvendinimu, įskaitant sistemos komponentų integravimą į vieną visumą.

Fizinė dekompozicija – tai sistemos architektūros skaidymas į komponentus*(fizines dalis)*, neatsižvelgiant kiek ir kokių funkcijų jie turės vykdyti. Skaidoma hierarchiškai iki neskaidžių bazinių elementų.

Funkcinė dekompozicija – tai sistemos skaidymas į komponentus pagal atliekamas funkcijas, neatsižvelgiant į fizines dalis ar kelios fizinės dalys šias funkcijas valdys.

Sistemos gyvavimo ciklo modelis – tai formalizuotas, *tvarkingas laike, projekto stadijų ir jų sąryšių*, aprašas, *kurį lengva suprasti ir paaiškinti kitiems*, suskaidantis visą darbą į mažus, aprėpiamus ir valdomus vienetus.

Sistemos įgyvendinamumas – tai matas, nusakantis ar sistemą įmanoma sukurti ir ar klientui verta ją kurti.

Sistemos įgyvendinamumo analizė – tai techninių ir ekonominių duomenų analizės procesas, atliekamas tikslu įvertinti sistemos įgyvendinamumo laipsnį.

Sistemos projekto atkūrimas – ta atvirkščiosios inžinerijos proceso dalis, kurios metu žinių apie dalykinę sritį ir išorinė informacijos pagalba, įvairiais samprotavimo būdais, iš nagrinėjamos sistemos analizės duomenų, bandoma išgauti aukštesnių lygių abstrakcijas, aprašančias sistemos sandarą ir veikimą.

Verslo procesas – tai tarpusavyje perpintų veiklų rinkinys, pradedamas vykdyti įvykus tam tikram įvykiui ir kuriantis konkrečius rezultatus, reikalingus užsakovui.

Verslo sistema – rinkinys tarpusavyje perpintų verslo procesų, vykdomų įmonės, siekiant tam tikrų(dažnai ilgalaikių) tikslų, tiesiogiai susijusių su tam tikrų gaminių gamyba/paslaugų tiekimu. Aprašo veiklas, reikalingas sukurti produktus bei teikiamas paslaugas.
Verslo sistema = verslo procesai + žmonės + informacija + technologijos.

Verslo inžinerija – tai inžinerinė metodika, įgalinanti apibrėžti, projektuoti ir kurti teisingai veikiančias verslo sistemas. Ji apima verslo procesų ir jiems vykdyti reikalingų resursų analizę, specifikavimą ir modeliavimą.

Verslo proceso reinžinerija – tai esminis verslo proceso pergalvojimas ir radikalus perprojektavimas siekiant reikšmingo to proceso našumo pagerinimo, *dažnai apimantis daugelį valdymo lygmenų, pradedant nuo tikslų ir rezultatų, o ne nuo atliekamų veiksmų.*

Projektas – tai griežtai apibrėžtą tikslą turintis procesas, su tiksliai nustatytomis pradžios ir pabaigos datomis, nustatytu užduočių rinkiniu ir fiksuotu biudžetu.

Projekto vadyba – tai resursų planavimo, naudojimo, valdymo ir kontrolės procesas, užtikrinantis projekto įgyvendinimą laiku, neviršijant biudžeto ir tenkinant reikalavimus. Ja siekiama sukurti/suskaidyti kuriamą sistemą į komponentus*(aparatūrą, PĮ ir kt.)* ir kiekvienam jų priskirti atitinkamas užduotis ar jų grupes.

Užduotis*(projekto)* – tai griežtai apibrėžtos apimties darbas, duodantis griežtai nustatytus rezultatus, pavestas atlikti, *vienam arba keliems*, projektą vykdančios grupės nariams.

Rezultatai*(projekto)* – tai, ką privalo sukurti ir pateikti projektas*: dokumentai, produktai, paslaugos, procesai, savybės.*

Kontroliniai taškai – direktyvinės datos, kurioms turi būti gauti griežtai apibrėžti tarpiniai projekto rezultatai. Jie naudojami projekto progresui matuoti.

Sistemos gyvavimo ciklas – laikotarpis nuo sistemos sumanymo iki jos naudojimo pabaigos (demontavimo) momento. Jis skaidomas į stadijas (phases), kiekvienai iš kurių numatomi kontroliniai taškai.

Suinteresuotasis asmuo – tai fizinis/juridinis asmuo(ar jų grupė), besidomintis projekto išeiga, dėl to, kad:
- arba projekto rezultatai turės jam kokį nors (teigiamą/neigiamą) poveikį
- arba pats nori daryti projektui kokį nors poveikį.

Užsakovas – tai subjektas, kurio poreikius vykdant projektą yra siekiama patenkinti. Jis finansuoja projektą, bei vertina ir naudoja jo rezultatus.

Informacinė sistema – tai verslo sistemos posistemis, apimantis komunikacinius ir informacinius verslo aspektus. Tai rinkinys darniai veikiančių komponentų, *renkančių, apdorojančių, saugančių, surandančių ir paskleidžiančių* informaciją bei duomenis, palengvinančius verslo *planavimą, valdymą, koordinavimą ir analizę* bei verslui reikalingų sprendimų priėmimą.

IS inžinerija – tai inžinerinė metodika, įgalinanti apibrėžti, projektuoti ir kurti teisingai veikiančias IS.

IS kūrimo metodas – tai integruotas procedūrų, technikų, produkto aprašų ir įrankių rinkinys, skirtas efektyviai, produktyviai ir sklandžiai palaikyti technologinį IS inžinerijos procesą.

Programinė įranga – kompiuterinės programos bei kompiuterinės duomenų ir žinių bazės.

Programinis produktas – tai kompiuterinių programų*, jų naudojamų procedūrų, susijusių dokmentų ir duomenų*, rinkinys*(t.p. kiekvienas jo elementas)*, skirtas pateikti užsakovui arba galutiniam vartotojui.

Programų sistema – tai iš programinės įrangos komponentų sudaryta sistema.

Didelė programų sistema – tai PS turinti, ne mažiau kaip, 50 000 vykdomojo kodo eilučių.

Programos kūrimas - veiklų seka, pasibaigiančių kokio nors programinio produkto sukūrimu.

PS kūrimas - tai procesas, kuriuo:
- vartotojų poreikiai pertvarkomi į PS reikalavimus;
- PS reikalavimai paverčiami tos sistemos projektu, o jis transformuojamas į programos kodą;
- atliekamas to kodo testavimas, dokumentavimas ir sertifikavimas*(leidimas eksploatuoti).*

Testavimas – tai automatiškai vykdomų testų kūrimas, padedantis apsisaugoti nuo "*patobulinimų*" su nenumatytom pasekmėm.

PĮ testavimas – tai dinaminis PĮ elgsenos tikrinimas, atliekamas panaudojant baigtinį testų skaičių, ir tos elgsenos palyginimas su reikalavimų specifikacija numatyta elgsena.

Yra išskiriamos dvi pagrindinės PĮ testavimo rūšys:

Black-box testing - testuotojas mato tik vartotojo interfeisą, nežinodamas apie vidinę PS sandarą.

White-box testing – testavimui reikia žinių apie vidinę PS sandarą; testuojama remiantis vidine PS sandara.

Programinis komponentas – tai vykdomojo kodo blokas, realizuojantis vieną ar kelias paslaugas, prie kurių prieinama per atitinkamus interfeisus, kurie niekuomet nekeičiami, nors paslaugos realizacija ir, netgi, jos funkcionalumas gali būti keičiami.

Programavimo aplinka*(PE)* **– tai:**

- programavimo įrankiai ir jiems pritaikyta TĮ,
- darbo procedūros ir projekto duomenų bazės,
- programų bibliotekos,
- kitos progr. aplinkos priemonės bei procedūros, palaikančios realizuojamą programavimo technologiją.

Programiniai įrankiai –tai aparatūriškai ar programiškai realizuotos priemonės/programos, padedančios padidinti viso darbo našumą ar kitaip patobulinti tą PS ar kokios kitos PĮ kūrimo procesą.

PĮ kūrimo aplinka*(SDE)* – tai sudėtingos standartizuotas paslaugas teikiančios sistemos, palaikančios, *visas arba daugumą*, kokio nors technologinio proceso stadijų.

Integruota programavimo aplinka*(IDE)* – tai tarpusavyje integruotų įrankių visuma, dažniausiai vadoma specialios programos/sistemos, realizuotos kurios nors standartinės OS priemonėmis. Yra griežtai kontroliuojama ar visi įrankiai naudojami teisingai. *(Pvz. Java pupuliari IDE yra NetBeans, C++ populiari IDE yra CodeBlocks ir kt.)*

Įrankinės/programavimo(konstravimo) įrankiai – tai dokumentuotos programos, dažnai IDE kaip vienos iš galimybių suteikiama funkcija, bei kitos palengvinančios programavimą(rašyti, analizuoti, testuoti, keisti, saugoti kompiuteryje) taikomosios programos. *Pvz. Notepad++,NetBeans.*

Architektūriniu požiūriu skiriami du PS kūrimo aplinkų(PE) tipai:

Integruota projekto palaikymo aplinka*(IPSE)* **–** kuria bendrą projekto infrastruktūrą, į kurią leidžiama terpti įvairius programinius instrumentus. Palaiko ne tik PS kūrimo technologinį procesą, bet ir projekto valdymą, ir šios abiejų tipų veiklos yra tarpusavyje integruotos.

Kompiuterizuota PS inžinerija*(CASE)* **–** tai integruoti įrankių rinkiniai,suprojektuoti taip, kad palaikytų visas, *PS gyvavimo ciklo modelio numatytas*, užduotis ir visus, *toms užduotims vykdyti reikalingus*, procesus.

Masinė gamyba – galimybė efektyviai tiražuoti gaminį daugeliu egzempliorių. Masinė gamyba yra amato antitezė.

Standartas – tam tikras sektinas modelis, pavyzdys ar kokio nors dydžio matavimo būdas, nustatytas tam juridinę galią turinčios institucijos, papročiu ar visuotiniu susitarimu. Standartais vadina dokumentus, nustatančius privalomus ar rekomenduotinus reikalavimus, taisykles ar sąlygas, susijusius su ko nors našumu, projektavimo būdu, eksploatavimo būdu ar kokybės matavimu.

Standartizavimas – tai sąvokų, doktrinų, procedūrų ir konstrukcijų, *padedančių pasiekti reikiamą **suderinamumo, keičiamumo vieno kitu** ar **bendrumo** lygmenį gamyboje, eksploatavime, panaudojime, administravime ar kitoje srityje*, kūrimas ir įgyvendinimas.

Atvirieji standartai - kuriami specialiai tam skirtos organizacijos, kuriami bendradarbiaujant.

Firminiai standartai - kuriami vienos firmos viduje, užsidarius.

de jure - standartai, įteisinti kokiu nors juridinę galią turinčiu dokumentu.

de facto - dokumentais neįteisinti, bet praktikoje paplitę standartai.

Web sistema – tai WWW veikianti sistema, susidedanti iš *web serverio*, *web dokumentų* ir *web kliento*.

Web dokumentas – paprastai juo yra vadinamas HTML dokumentas.

Web klientas – tai PS, veikianti vartotojo AK, turinčiame *Web serverio* priegą. Dar vadinamas „naršykle“.

Objektų reikalavimų brokeris - tarpininkas tarp kliento ir komponentų su jų paslaugomis.

Interfeisų aprašymo kalba – tai kalba, kuria aprašomi komponentai, paslaugos ir jų interfeisai. Populiaresnės yra *C++ ir XML.*

Abstraktus duomenų tipas – tai matematinė duomenų ir ant jų apibrėžtų operacijų specifikacija. Jis abstraktūs ta prasme, kad dėmesys sutelkiamas į operacijas, ignoruojant realizavimo detales. Programoje, ADT pateikia interfeisus, maskuojančius jų realizavimą. *Tipiniai ADT pvz.: sąrašas, stekas, eilė, kompleksinis skaičius.*

Virtualiosios mašinos – tai programiškai realizuoti kompiuteriai, emuliuojami jūsų realaus kompiuterio.

Virtualioji mašina – tai abstrakcija, realizuojama panaudojant "*realią*" TĮ ir "*realią*" operacinę sistemą. Operacinė sistema*(Windows, Linux ir kt.)* taip pat yra virtualioji mašina.

Verslo procesų reinžinerija – tai verslo procesų apgalvojimas iš naujo ir toks jų radikalus perprojektavimas, kad iš esmės pakistų tokie verslo parametrai kaip kaštai, kokybė, darbų trukmė ir pan.

Trasavimas*(versijavimas)* – tai reikalavimas, įvardinamas vienareikšmiškai*(pvz., turi unikalų numerį)* ir turintis nuorodą į savo šaltinį. Skirtas visų pakeitimų stebėjimui tarp senesnės ir naujesnės versijos.

Operacinis poreikis*(vartotojo reikalavimas)* – tai skaičiavimo arba informacinė paslauga*(dalykinė programa, duomenų bazė, interneto tinklalapis ar kt.),* padedanti vartotojams pasiekti numatytus operacinius tikslus. Operacinius poreikius galima traktuoti kaip pačius bendriausius PS reikalavimus.

Pagrindinis RI uždavinys – transformuoti operacinius poreikius *(vartotojų reikalavimus)* į PS reikalavimus**.**

Svarbiausias PS kūrimo žingsnis – perteikti vartotojų*(kam sistema kuriama)* operacinius poreikius PS inžinieriams*(kas sistemą kurs.)*.

Verslo poreikių specifikacija – tai dokumentas, aprašantis:

- sistemos paskirtį*(verslo terminais)*;
- operacinius reikalavimus*(poreikius)*;
- planuojamą sistemos naudojimo scenarijų;
- sistemos aplinką*(operacinę ir aptarnavimo)*;
- pirminę sistemos įgyvendinamumo analizę.

Siekis – kokia užsakovo instancijos*(organizacijos/įmonės)* paskirtis? Ko ji siekia?

Vizija – kokia mūsų užsakovo instancija*(organizacija)* norėtų būti po 5-10 metų? Iš šio aprašo išplaukia ir būsimos sistemos vizija, nusakanti tos sistemos paskirtį ir kūrimo tikslus*(verslo terminais)*.

Strategija – darnus rinkinys tarpusavyje integruotų veiksmų, kuriais siekiama konkurencinėje kovoje maksimaliai išnaudoti turimus pranašumus.

Strategija – priemonė, principų rinkinys, kuriais vadovaujantis siekiama išpildyti viziją.

Verslo galimybės – pozityvios išorinės tendencijos/veiksniai, galinčios pagerinti verslo proceso efektyvumą, padėti pasiekti tikslus.

Verslo grėsmės – negatyvios išorinės tendencijos/veiksniai, galinčios trukdyti verslo proceso efektyvumui, ar siekiant tikslų.

Tikslų medis – tai medis, pavaizduojantis pasirinktos strategijos strateginius tikslus bei iš jų išplaukiančius operacinius tikslus. Skaitant medį nuo šaknies iki lapų, išreiškiama, kaip tikslai yra pasiekiami; skaitant nuo lapų iki viršūnės – kodėl tokie operaciniai tikslai pasirinkti.

Sistemos naudojimo diagrama – tai užduočių diagrama, papildyta komentarais, aprašančiais kokiu mastu kuriama PS palaikys kiekvienos verslo užduoties vykdymą.

Sistemos teikiamos naudos diagrama – tai užduočių diagrama, papildyta komentarais, aprašančiais ką išloš kiekvienas iš vartotojų, įdiegus kuriamą PS.

Žmonės

17. Kas tokia buvo Ada Lovelace?

Pirmoji programuotoja. Išrado paprogrames, ciklus.

18. Kas toks buvo Alan Turing?

Sukūrė pirmąjį matematinį kompiuterio modulį.

19. Kas toks buvo John von Neumann?

Šiuolaikinių kompiuterių architektūros formuotojas.

20. Kas tokia buvo Grace Hopper?

Sukurė 1-ąjį kompiliatorių*(progr. kalbos)*. 1-oji vieno iš pirmųjų elektroninių kompiuterių „*Mark I*“ programuotoja. Viceadmirolė, viena iš kompiuterių pionierių.

21. Kas toks buvo Edsger Dijkstra?

Sukūrė teorinius programavimo pagrindus. Algoritmų teorijos, struktūrinio programavimo pradininkas. Vienas iš pirmosios PSI knygos autorių. Laiško *„Go To statemenent considered harmful“* autorius.

22. Kuo pagarsėjo Donald Knut?

TeX kūrėjas. „*The Art Of Computer Programming*“ knygos autorius.

23. Kuo pagarsėjo Niclaus Wirth?

Pascal autorius(1968-72). Parašė pirmąją knygą programavimo metodikos klausimais. Modula, Oberon kūrėjas.

24. Kuo pagarsėjo Frederick Brooks?

Operacinės sistemos IBM OS/360 konstruktorius. PS kūrimo projektų vadybos teorijos pradininkas.

25. Kuo pagarsėjo Barry Boehm?

1-asis išpranašavo, kad išlaidos programinei įrangai peraugs išlaidas aparatūrai, sukūrė „*COCOMO*“ metodiką ir Spiralinį gyvavimo ciklo modelį. Žinomas dėl savo darbų apie PS kūrimo ekonominius aspektus.

26. Kuo pagarsėjo Alan Kay?

Išrado laptopus, sukūrė šiuolaikinių langų bei grafinio vartotojo interfeiso koncepciją. Žinomais darbais apie PK, Smalltalk*(progr. kalbą)* ir OP.

27. Kuo pagarsėjo Adele Goldberg?

Įnašu į Smalltalk, objektinę analizę ir objektinį projektavimą.

28. Kuo pagarsėjo Ivar Jacobson?

Darbais apie UML ir RUP, komponentus ir komponentines architektūras, užduotis*(use cases)*, verslo inžineriją.

29. Kuo pagarsėjo Grady Booch?

Įnašu į UML ir RUP, „*Booch*“ metodo autorius, aprašyto jo knygoje „*Object Oriented Analysis And Design*“.

30. Kuo pagarsėjo James Rumbaugh?

Įnašu į UML ir OMT, vienas iš duomenų srautų architektūros kompiuterio išradėjų. Vienas iš žymiausių PĮ kūrimo metodikos specialistų pasaulyje.

31. Kas tokie vadinami trim draugais *(Three Amigo)*? Kuo garsūs?

Sukūrė pirmąjį UML variantą. Trys draugai – tai Grady Booch, Ivar Jacobson ir James Rumbaugh, daug metų paskyrę projektavimo kalboms unifikuoti.

32. Kuo pagarsėjo David Parnas?

Vienas iš žymiausių šių dienų informatikų. Modulinio projektavimo teorijos pradininkas. Viešai pasisakė prieš Reigano žvaigždžių karo iniciatyvą, teigdamas, kad kol kas neįmanoma sukurti patikimą tokio sudėtingumo PĮ.

33. Kas toks yra Bill Gates?

William H. (Bill) Gates (1955) – Microsoft įkūrėjas (1975), vienas iš pačių turtingiausių pasaulio žmonių. Programuoja nuo 13 metų. Profesinę karjerą pradėjo studijuodamas Harvardo universitete, kur jis parašė Basic kalbos kompiliatorių MITS Altair kompiuteriui (1973). ~~Universiteto nebaigė.~~ Universitetą baigė tik 2007 m.

34. Kuo pagarsėjo Alistair Cockburn?

Turėjo 16 metų kompiuterių inžinierius darbo patirtį, kai IBM paprašė jį parengti objektinių projektų vykdymo metodiką. Pastaruosius 10 metų užsiima PSI klausimais, pagarsėjo savo pasiūlyta užduočių modeliavimo metodika. Parašė knygą apie judriuosius PS kūrimo metodus.

35. Kas tokie vadinami keturių gauja (*Gang Of Four*)? Kuo ji garsi?

Taip buvo vadinama tam tikra politinė grupuotė maoistinėje Kinijoje. Mėgdžiojant pavadinimą, taip vadinami garsiosios „*Design Patterns*“ knygos autoriai: E. Gamma, R. Helm, R. Johnson, J. Vlissides.

Pap 1. Kas yra Dennis Ritchie? Kuo garsus?

C kalbos autorius(1969-73). Kartu su Ken Thompson, 1971 m. sukūrė UNIX. C – pirmoji nuo kompiuterio architektūros nepriklausanti mašininio lygmens programavimo kalba.

Pap 2. Kas yra Richard Stallman ? Kuo garsus?

Vadinamas „geriausiu pasaulio programuotoju“. GNU ir Freeware judėjimo iniciatorius(1984), GNU įrankio g++ autorius. Šiuo metu g++ LINUX

Pap 3. Kas yra James Martin ? Kuo garsus?

101 vadovėlio autorius. Išpranašavo e-verslo sistemas. Pulitzerio premijos laureatas. Vienas iš didžiausių autoritetų informatikos srityje.

Pap 4. Kas yra Juozas Zalatorius? Kuo garsus?

Pirmosios Lietuvoje PĮ kuriančios firmos “*Bitas*”(1989) įkūrėjas ir direktorius. Vėliau firma peraugo į „*Baltic Amadeus*“. Vienas iš asociacijos “*Infobaltas*” kūrėjų.

Programinės įrangos rūšys

29. Programinės įrangos rūšys:

- Sisteminė PĮ
- Taikomoji(dalykinė) PĮ
- Užsakomoji PĮ
- Visuotinio naudojimo PĮ
- Laisvai platinama PĮ *(shareware)*
- Nemokama PĮ *(freeware)*
- Įterptoji PĮ *(embedded software)*
- Kritinė PĮ
- Realiuoju laiku veikianti PĮ
- Duomenų apdorojimo PĮ
- Grupinio naudojimo PĮ *(groupware)*
- Tarpinė PĮ *(middleware)*
- Ribinė PĮ *(front-end software)*
- Užslėptoji PĮ *(back-end software)*
- Aparatūrinė PĮ *(firmware)*
- Komponentinė PĮ *(componentware)*

30. Kokia PĮ vadinama *sistemine programine įranga*? (1)

Sisteminė PĮ – apima operacines sistemas, pagalbines programas, tvarkančias kompiuterių resursus. Sistemos PĮ sudaro programos, esančios tiesioginėje saveikoje su aparatūra.

31. Kokia PĮ vadinama *taikomaja programine įranga*? (1)

Taikomoji*(dalykinė)* PĮ – programa ar jų rinkinys, suprojektuoti spręsti dalykinio pobūdžio problemas, kylančias galutiniems vartotojams. Tai gali būti DB valdymo programos, tekstų redagavimo programos, grafikos apdorojimo programos ir kt. Taikomoji PĮ negali veikti be sisteminės PĮ (*tuo pačiu, be aparatūros)*.

32. Kuo skiriasi *užsakomoji* ir *visuotino naudojimo* PĮ? (1)

Užsakomoji PĮ kuriama kokrečiam užsakovui pagal jo poreikius, visuotino naudojimo PĮ parduodama visiems vienoda.

33. Kuo skiriasi *laisvai platinama* ir *nemokama* PĮ? (1)

Laisvai platinama PĮ paprastai yra mokama, skirta išbandymui ir neužsimokėjus po 2-4 savaičių nustoja veikti ar apribojamos jos funkcijos. Tuo tarpu nemokama PĮ galima naudotis neribotai.

34. Kokia PĮ vadinama *įmontuota programine įranga*? įmontuota programų sistema? (2)

Įterptoji*(įmontuotoji)* PĮ*(embedded software)* - į aparatūrą*(pvz. automobilį)* montuojama PĮ. Sunkiai keičiama.

35. Kokia PĮ vadinama *kritine programine įranga*? Pateikite pavyzdžių (1)

Tokia, kurios trykiai padaro didžiulę žalą. *Pvz.: valdanti branduolinį reaktorių, naudojama elektros tinkluose.*

36. Kokie yra *užsakomosios*, *visuotino naudojimo* ir *įmontuotosios* PĮ skirtumai? (3)

- ***Užsakomoji PĮ*** - mažas tiražas, maži jos vykdymui išskirti CPU resursai*(pasaulio mastu)*, didelės darbo sanaudos PĮ sukurti*(pasaulio mastu)*.
- ***Visuotinio naudojimo PĮ*** - vidutinis tiražas, dideli CPU resursai, vidutinės darbo sanaudos PĮ sukurti.
- ***Įmontuotoji PĮ*** - didelis tiražas, maži procesoriaus resursai, mažos darbo sanaudos PĮ sukurti.

37. Kokia PĮ vadinama *realaus laiko masteliu veikiančia programine įranga*? realaus laiko programų sistema? (1)

- Realaus laiko masteliu veikiančia PĮ vadinama programinė įranga, stebinti ir valdanti technines sistemas, kur turi būti griežtai fiksuotas reakcijos laikas ir didžiausia problema - saugumas.
- Realaus laiko programų sistema - sistema, reaguojanti į išorinio pasaulio įvykius per griežtai nustatytą (paprastai labai trumpą) laiką. Dažniausiai realaus laiko sistemos stebi arba valdo kokius nors įrenginius arba kokį nors išorinį procesą, vykstantį lygiagrečiai su programų sistemos veikimu.

38. Kokia PĮ vadinama *duomenų apdorojimo programine įranga*? (1)

Ta, kuri naudojama versle, o jos svarbiausios problemos yra skaičiavimų tikslumas ir duomenų saugumas.

39. Kokia PĮ vadinama *grupinio naudojimo programine įranga*? (1)

Ta, kuri padeda organizuoti grupinį darbą kompiuterių tinkle*(dažniausiai, organizacijos tinkle)*.

40. Kokia PĮ vadinama *tarpine programine įranga*? (1)

Skirta organizuoti kelių skirtingų rūšių PĮ sąveikai, įskaitant informacijos mainus. Pvz, sąveikai tarp taikomosios PĮ ir operacinės sistemos*(sisteminės PĮ)*.

41. Kuo skiriasi *ribinė* ir *užslėptoji* PĮ? (1)

- **Ribinė PĮ** - tiesiogiai pasiekiama vartotojo, kuria jam prieigą prie resursų ir taikomosios PĮ.
- **Užslėptoji PĮ** - tiesiogiai nepasiekiamas funkcionalumas, slypintis už taikomųjų programų.
    - Pvz.: duomenų bazės serveris.

42. Kokia PĮ vadinama *aparatūrine programine įranga*? (1)

Tokia, kuri saugoma nuo kompiuterio operatyviosios atminties nepriklausančiu būdu, paprastai, kaip mikroprogramos, įrašytos į ROM*(skaitomąją atmintį)*. Įprastai, keisti aparatūrinės PĮ, vartotojams neleidžiama.

Programų sistemų rūšys

46. Programų sistemų rūšys: (2)

- Centralizuotos sistemos
- Decentralizuotos sistemos
- Autonominės sistemos
- Integruotosios sistemos
- Didžiųjų ESM sistemos *(mainframe systems)*
- Paketinės sistemos *(batch systems)*
- Tiesioginės prieigos sistemos *(on-line systems)*
- Interaktyviosios*(dialoginės)* sistemos *(interactive systems)*
- Realaus laiko sistemos *(real-time systems)*
- Išskirstyta sistema *(distributed systems)*
- Personalinių kompiuterių sistemos *(PC systems)*
- Kliento serverio sistemos *(client-server systems)*
- Lygiųjų sistemos *(peer-to-peer systems)*
- Web sistemos *(Web systems)*
- Įmontuotosios*(įterptosios)* sistemos *(embedded systems)*

47. Kuo skiriasi *centralizuotos* ir *decentralizuotos* PS? (1)

Centralizuotose sistemose visi resursai ir valdymas yra centralizuoti. Decentralizuotose nėra išskirto komponento, administruojančio visos sistemos darbą.

48. Kokios PS vadinamos *autonominėmis*? (1)

Tokios, kurios gali veikti vienos pačios, nes visi reikiami resursai yra jų viduje.

49. Kokios PS vadinamos *integruotosiomis*? (1)

Tokios, kuriose skirtingos dalykinės paskirties programos*(kurios paprastai veikia kaip autonominės sistemos),* nuo pat pradžios yra projektuojamos ir kuriamos taip, kad veiktų vienos sistemos sudėtyje.

50. Kokios PS vadinamos *didžiųjų ESM sistemomis*? *personalinių kompiuterių sistemomis*? (2)

Didžiųjų ESM sistema - sistema, veikianti kokioje nors didžiojoje ESM arba tokių kompiuterių tinkle. Vartotojai naudojasi tokia sistema per lokalius ar nutolusius terminalus, valdomus didžiosios ESM ir leidžiančius rašyti į ją duomenis, juos keisti bei peržiūrinėti.

PK sistemos - sistemos, kurioms veikti pakanka personalinių kompiuterių teikiamų galimybių.

51. Kokios PS vadinamos *paketinėmis sistemomis*? (1)

Tokios, kurioms pradėjus veikti, yra nereikalingas joks vartotojo įsikišimas. Jos veikia automatiškai.
Pvz.: bankuose paketiniu režimu kiekvieną dieną yra apdorojami įrašai apie tą dieną vykdytas operacijas.

52. Kokios PS vadinamos *tiesioginės prieigos sistemomis*? (1)

Tos, kurios prijungtos prie tinklo arba per tinklą ir jomis vartotojas gali naudotis tiesiogiai. Pvz., duom. bazės.

53. Kokios PS vadinamos *išskirstytomis sistemomis*? (1)

Tai išskirstytos kompiuterių tinkle PS, t.y. veikiančios keliuose kompiuteriuose iš karto, o atliekamas darbas yra paskirstomas tarp kelių kompiuterių. Tai gali būti didžiosios ESM, PK arba abiejų tipų kompiuterių mišinys.

54. Kokios PS vadinamos *kliento serverio sistemomis*? (2)

Tai išskirstytų PS klasė. Tokiose sistemose išskirstytus skaičiavimus atlieka dvi pagrindinės tarpusavyje tinklu susietų komponentų rūšys - klientai ir serveriai.

Klientai veikia prie tinklo prijungtuose kompiuteriuose, naudojamuose prieigai prie bendro naudojimo resursų, dažniausiai personaliniuose kompiuteriuose.

Serveriai veikia specializuotuose kompiuteriuose, vadinamuose serveriniais kompiuteriais, prijungtuose prie to pačio tinklo ir teikiančiuose klientams tam tikras paslaugas.

55. Kokios PS vadinamos *lygiųjų sistemomis*? (1)

Tai sistemos, kuriose informacijos vartotojai tuo pačiu yra ir informacijos tiekėjai. Informacijos mainai paprastai vyksta tarp dviejų partnerių. Dar daugiau, partneriai gali atsirasti ir dingti, o sistema bet kuriuo momentu gali užprašyti informacijos iš bet kurio tuo momentu prieinamo partnerio.

56. Kokios PS vadinamos *Web sistemomis*? (2)

Web serverio PĮ veikia serveriniame kompiuteryje ir vykdo keletą funkcijų, įskaitant:

- interfeiso su tinklu organizavimą
- dokumentų saugojimą
- interfeiso su išorinėmis programomis organizavimą

Programų sistemų inžinerijos pareigybės

Pagrindinės pareigybės(vaidmenys), esančios kuriant PS pramoniniu būdu: (2)

- Sisteminis analitikas
- Sistemos architektas
- Posistemio architektas*(posistemio projektuotojas)*
- Inžinierius programuotojas
- Testuotojas
- Sistemų integruotojas
- Projekto vadovas
- Produkto inžinierius
- Proceso inžinierius*(technologas)*,
- Informacijos inžinierius,
- Diegimo inžinierius,
- Sistemos administratorius

Už ką yra atsakingas sisteminis analitikas ir ką jis dirba? (2)

- Atsakingas už užsakovo poreikių nustatymą, analizę ir transformavimą į PS reikalavimus
- Privalo gerai išmanyti dalykinę sritį
- Privalo būti įvaldęs sisteminės analizės metodus
- Privalo gerai mokėti UML

Pagrindiniai darbai: intervių ėmimas, funkcinių ir nefunkcinių poreikių nustatymas, dalykinės srities modeliavimas, sistemos reikalavimų formulavimas ir analizė, tarpininkavimas tarp užsakovo ir sistemą kuriančio inžinierinio personalo

Už ką yra atsakingas sistemos architektas ir ką jis dirba?(2)

Iš sistemos architekto reikalaujama:

- patirties (*tiek dalykinėje srityje, tiek ir PS kūrimo)*; lyderio savybių *(gebėjimo susitelkti, pasitikėjimo savimi, charizmos, autoriteto)*; gebėjimo bendrauti ir užsakovo, ir inžinierinio personalo terminais; aktyvumo ir gebėjimo siekti užsibrėžtų tikslų *(priimti sprendimus, nebijoti rizikos, laiku užbaigti darbą)*, gero UML žinojimo
- Pageidautina, kad architektas būtų geras programuotojas!

Pareigos:

- sistemos architektūros, įgyvendinančios sistemos reikalavimus, parinkimas ir projektavimas; dalyvavimas sistemos peržiūrose ir inspektavime; techninis vadovavimas visam sistemą kuriančiam inžineriniam personalui

Už ką yra atsakingas PS posistemio architektas ir ką jis dirba? (2)

- Asmuo, atsakingas už kokio nors posistemio įgyvendinimą;
- Veikia kaip tarpininkas tarp sistemos architekto ir posistemį kuriančių PS inžinierių;
- Privalo gerai mokėti UML;
- Yra pasiruošęs, reikalui esant, veikti sistemos architekto ar inžinieriaus programuotojo vaidmenyje.

Pareigos: eskizinis posistemio projektavimas: posistemio skaidymas į klases, klasių klasterizavimas, klasių hierarchijų projektavimas, interfeisų projektavimas, posistemio skaidymas į versijas, testų projektavimas ir t.t.

Už ką yra atsakingas inžinierius programuotojas ir ką jis dirba?(2)

- Atsakingas už detalųjį modulių projektavimą, kodavimą ir testavimą
- Dalyvauja rengiant projektinius dokumentus ir vartotojui skirtą dokumentaciją
- Privalo mokėti UML ir labai gerai mokėti programavimo kalbas
- Dideliuose projektuose specializuojasi: interfeisų programavimas, web programos, saugos inžinerija, įrenginių valdymas, probleminis programavimas ir t.t.

Už ką yra atsakingas testuotojas ir ką jis dirba?(2)

- Viena iš nedėkingiausių pareigybių:
  - programuotojai juos traktuoja kaip studentai egzaminus, t.y. kliūtis, kurias reikia kažkaip apeiti
  - vadovai kaip (perteklinius) negamybinės sferos darbuotojus, kuriems dar tenka ir pataikauti
- Testuotojas privalo būti labai objektyvus ir tvarkingas

Pareigos: Testavimo planavimas ir vykdymas, testų konstravimas ir priežiūra, baigiamieji bandymai; kiekybinių kokybės vertinimo kriterijų formulavimas

Už ką yra atsakingas sistemų integruotojas ir ką jis dirba? (2)

- Surenka sistemos komponentus į vieną visumą, daro užsakovui atiduodamos sistemos ruošinius;
- Privalo būti gerai įsisavinęs visus projektui kurti naudojamus įrankius;
- Privalo būti pajėgus kurį laiką*(savaitę-dvi)* dirbti labai intensyviai*(įskaitant savaitgalius ir naktis!)*

Už ką yra atsakingas produkto inžinierius ir ką jis dirba?(2)

- Užsiima sistemos konfigūracijos valdymu*(procesas, vykstantis visą projekto vykdymo laiką)*
- Privalo turėti buhalterio įgūdžius ir būti labai tvarkingas bei atsakingas asmuo
- Daugumoje projektų produkto inžinieriaus ir integruotojo pareigas vykdo tas pats asmuo
- Mažuose projektuose jis gali vykdyti ir testuotojo pareigas

Už ką yra atsakingas projekto technologas ir ką jis dirba?(2)

- Projektuoja technologinį PS kūrimo procesą, komplektuoja, pritaiko ir prižiūri programinę projekto vykdymo aplinką, konsultuoja jos panaudojimo klausimais projekto vykdytojus
- Turi gerai išmanyti programuotojų ir PS inžinierių darbą bei būti susipažinęs su visa instrumentinių priemonių gausa

Už ką yra atsakingas diegimo inžinierius ir ką jis dirba? (2)

Atsako už sistemos perdavimą užsakovams, instaliavimą, bandomosios versijos priežiūrą, jos metu išryškėjusių klaidų bei trūkumų šalinimą, vartotojų mokymą ir konsultavimą, visus kitus PS diegimo darbus.

Už ką yra atsakingas informacijos inžinierius ir ką jis dirba?(2)

- Rengia vartotojui skirtą dokumentaciją *(vadovus, help-failus, demo-pavyzdžius)* ir sistemos vartotojams pateikiamų pranešimų tekstus, projektuoja formas, ataskaitas, užduočių formulavimo kalbas
- Privalo gerai mokėti kalbą, žinoti tiek informatikos, tiek ir dalykinės srities terminiją, mokėti sklandžiai, glaustai ir prasmingai dėstyti savo mintis, gebėti perteikti sudėtingas sąvokas paprastais žodžiais
- Turi išmanyti vartotojo psichologiją.
- Gebėti bendrauti su inžineriniu personalu, „*iškvosti*" viską, ką apie sistemą reikia pateikti vartotojui
- Dažniausiai dirba keliuose projektuose iš karto

Už ką yra atsakingas sistemos administratorius ir ką jis dirba?(2)

Atsako už projektui naudojamų kompiuterinių resursų *(TĮ, PĮ, tinklai ir kt.)* aptarnavimą, priežiūrą ir apsaugą

- Dideliuose projektuose apsauga gali užsiimti spec. žmogus – saugumo inžinierius*(security engineer)*
- Dažniausiai, sistemos administratorius aptarnauja visus organizacijos vykdomus projektus
- Sistemos administratorius turi dirbti tolygiai su proceso inžinieriumi/dažnai tai yra tas pats asmuo

Programų sistemų inžinerijos principai

27. Programų sistemų inžinerijos principai:

- turinių atskyrimo *(concern separation)* principas,
- dekompozicijos principas,
- juodosios dėžės *(black box)* principas,
- abstrakcijos principas,
- unifikavimo *(uniformity)* principas,
- struktūrizavimo principas,
- sistemų atvirumo principas,
- interfeisų komfortiškumo principas,
- metaforizavimo principas,
- reaktyvumo *(agency)* principas

28. Paaiškinkite kas tai yra turinių atskyrimo principas. (3)

Turinių atskyrimo principo esmę sudaro griežtas visa ko klasifikavimas ir tinkamas klasifikavimo požymių parinkimas. Neįvaldžius turinių atskyrimo principo, neįmanomi nei jokia inžinerinė veikla, nei joks tvarkingas mastymas apskritai. Taisyklė:

- PS turi būti taip projektuojamos ir konstruojamos, kad dalykinės srities konceptai turėtų sistemoje tiesioginius atitikmenis. Sistemos dalių sąveika turi tiksliai atspindėti tų konceptų ryšius
    - tai reiškia, kad PS turi būti kuriama remiantis atitinkamos dalykinės srities (verslo) koncepciniu modeliu,
    - ši taisyklė iš tiesų yra dalinis turinių atskyrimo principo atvejis, ji yra dar vadinama konceptualizavimo principu,
    - bendruoju atveju turinių atskyrimo principo formuluotė yra gerokai bendresnė.
- Konceptualizavimo principą pasiūlė Edward Yourdan ir Larry L. Constantine knygoje Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design (1979)
    - vadovaujantis šiuo principu, yra ženkliai sumažinamos pastangos, reikalingos PS realizuoti, prižiūrėti ir perdaryti.
    - PS realizavimo kaštai yra mažesni tuomet, kai sprendžiamos problemos dalys yra: aiškiai atskirtos viena nuo kitos, aprėpiamos,sprendžiamos nepriklausomai viena nuo kitos.
    - PS priežiūra supaprastėja tuomet, kuomet tos sistemos dalys: atitinka sprendžiamos problemos dalis,yra aprėpiamos,gali būti keičiamos nepriklausomai viena nuo kitos.
    - Sistemos reinžinerija supaprastėja, kuomet:sistemos struktūra atitinka sprendžiamos problemos struktūrą,sistemos dalis galima keisti nepriklausomai viena nuo kitos.

29. Paaiškinkite kas tai yra dekompozicijos principas. (3)

Dekompozicija – tai procesas, kuriuo didelės problemos suskaidomos į mažesnes, lengviau aprėpiamas dalines problemas.

Taisyklės:

- Visus sudėtingus PS inžinerijos procesais kuriamus objektus reikia skaidyti į kiek galima paprastesnes sudėtines dali (modulius) ir procesą, kuriuo kuriamas objektas, taikyti kiekvienam moduliui atskirai.
    - Toks skaidymas vadinamas dekomponavimu. Nagrinėjamo proceso požiūriu visi moduliai turi būti vieno lygmens. Jie taip pat turi būti nepriklausomi, t.y. tokie, kad, atlikus norimus veiksmus su kiekvienu iš jų atskirai ir sujungus tokiu būdu gautus rezultatus į vieną visumą, būtų gautas pageidaujamas galutinis rezultatas.
    - Procesas turi būti kartojamas tol, kol objektas nėra suskaidytas į primityvus, t.y. lengvai suvokiamas ir neskaidžias dalis
- Dalys vadinamos neskaidžiomis, jei jų negalima toliau skaidyti, nepažeidžiant jų savarankiškumo
    - Programuojant C kalba, sistema dekomponuojama į funkcijas ir abstrakčius duomenų tipus. Programuojant C++ arba Java kalba, sistema dekomponuojama į klases.
- Problemos dalys turi būti kiek galima labiau nepriklausomos (turinių atskyrimo principas)
- Norint pritaikyti dekompozicijos principą, būtina atsakyti į šiuos klausimus:
- Stipriai perpintos problemos dalys turi būti realizuojamos vienoje sistemos dalyje, nesusijusios problemos dalys – skirtingose dalyse.
- Sistema organizuojama taip, kad nei viena dalis nebūtų didesnė, nei būtina tai problemos daliai išspręsti.
- Sistemą būtina taip organizuoti, kad tarp jos dalys nebūtų siejamos tokiais ryšiais, kurie neatitinka problemos dalis siejančių ryšių.

Kuriant dideles PS, reikia:

- jas dekomponuoti į komponentus, specifikuoti kiekvieno interfeiso elgseną ir jo interfeisą, kiekvieną komponentą kurti ir testuoti atskirai; komponentus sujungti į bendrą sistemą tiktai po to, kai jie visi yra baigti testuoti.
- ***Privalumai:*** Programuotojas užsiiminėja vieno komponento detaliu projektavimu. Nuošalyje lieka dešimtys kitų komponentų su šimtais savo detalių, apie kurias jam nereikia galvoti projektuojant ir programuojant savąjį komponentą. Tai kitų programuotojų rūpestis.

30. Paaiškinkite kas tai yra juodosios dėžės principas. (3)

Juodoji dėžė – tai sistema/ komponentas, apie kurią yra žinoma: *įeiga*, *išeiga*, *kaip išeiga priklauso nuo įeigos*, bet nežinoma kas vyksta jos viduje. Ja galima naudotis nieko nežinant apie realizaciją! Tai paprasčiausia iš visų abstrakcijų, kuria galima aprašyti sistemos veikimą, nieko nesakant apie jos realizavimą.

- Programiškai jas galima suskirstyti į dvi grupes: 1. **statines**, 2.**dinamines**.
- Šis principas įgalina paprastai ir išsamiai specifikuoti modulius, nurodant pradinius duomenis ir jų šaltinius (iš kur imti) bei rezultatus ir jų talpyklas (kur padėti)
- Projektuojant, jomis galima naudotis kaip koncepcinio lygmens sistemos komponentais.
- Projektuojant prisireikus kokios nors funkcijos/savybės, ją galima apibrėžti kaip „juodąją dėžę“ ir ja naudotis, nesirūpinant kaip ji bus realizuota
- Įeigai turi priklausyti tik tai, ko tikrai reikia išeigai gauti.
- Pagrindinis testas: ar paprasta nusakyti juodosios dėžės paskirtį?
- Funkcinėms dėžėms: ar galima aprašyt jos paskirtį veiksmažodžiu*(dėžės pavadinimas)*, o įeiga/išeiga daiktavardžiais*(dėžės parametrai)* ?
- Objektinėms dėžėms: kas įvyks su objektu, gavus bet kurį dėžės apdorojamą pranešimą.
- Kviečiantysis kviečiamąjį modulį visuomet turi traktuoti kaip juodąją dėžę!

Informacijos maskavimo *(information hiding)* principas yra dalinis juodosios dėžės principo atvejis*(„private“ metodai, inkapusliacija)*.

31. Paaiškinkite kas tai yra abstrakcijos principas. (3)

Sudėtingos PS yra projektuojamos*(o kartais ir realizuojamos)* sluoksniais. Abstrakcija gaunama ignoruojant neesmines detales, paliekant tiktai esminius ko nors aspektus.

Abstrahavimasis – tai apibendrinimo procesas. Jo metu ieškoma panašumų tarp daiktų. Abstrakcijos gaunamos jungiant panašius daiktus į grupes.

- ***Taisyklė:*** Kiekvieną PSI procesą, kiekvienam juo kuriamam objektui taikyk iteratyviai, kiekvienoje iteracijoje nagrinėdamas objektą vis didesniu detalumu, ignoruojant nesvarbias tame lygmenyje.
- Bet kuri sistema turi keletą abstrakcijos lygmenų
- PS abstrakcijos lygmenys dažnai yra gretinami su modulių lygmenimis*(t.y. lieka veikiančioje sistemoje)*
- Kartais abstrakcijos lygiai esti tik projekto dokumentacijoje*(t.y. veikiančioje sistemoje jų nebelieka)*.

32. Paaiškinkite kas tai yra pažingsninių patikslinimų principas. (3)

Nuoseklių tikslinimų*(stepwise refinement)* principas yra specialus abstrakcijos principo atvejis.

- Nuoseklūs patikslinimai - tai pažingsninis procesas, kurį taikant yra imama programos*(modulio)* funkcija ir dekomponuojama į subfunkcijas. Tęsiant šį procesą, galų gale yra gaunami reikiamos programos*(modulio)* kodo operatoriai.
- Tai iteratyvus procesas, kurio kiekviena iteracija turime vis detalesnį ir detalesnį programos tekstą.
- Dekomponuodami į vis smulkesnes ir smulkesnes subfunkcijas, mes vis tikslau ir tiksliau užrašome sprendžiamos problemos sprendimo algoritmą, t.y. žingsnis po žingsnio atliekame nuoseklius algoritmo tikslinimus.
- Didelės ir sudėtingos programos gali turėti daug tikslinimo lygmenų
    - Pradiniai patikslinimai yra labai bendri ir ne daug ką skiriasi nuo specifikacijos, vėlesni yra vis labiau ir labiau panašūs į galutinį programos kodą
    - Tarpiniuose lygmenyse operatoriai atitinka tam tikrus programus gabalus, bet ir jie dar yra nepakankamai detalizuoti, kad juos būtų galima užrašyti kokia nors programavimo kalba.
    - Žemiausiojo lygmens operatoriai yra transformuojami į kokios nors programavimo kalbos operatorius (arba jų sekas).

Taisyklė:

1. Rasti bendriausią problemos sprendimo algoritmą. Jis turi būti sudarytas vadovaujantis funkciniu požiūriu, t.y. išreikštas užduočių, kurias atlikus bus gautas norimas rezultatas, terminais.
2. Po to, žingsnis po žingsnio, "tikslink" pasiūlytą algoritmą, dekomponuodamas užduotis į použduotis. Tikslinimas tęsiamas tol, kol galų gale použduotys yra išreiškiamos pasirinktąja programavimo kalba.

Tikslinimo procesas susideda iš 2 dalių:

- ***Analizės:*** kiekviename žingsnyje problemos sprendimo būdas analizuojamas vis didesniu detalumu;
- ***Sintezės:*** lygiagrečiai analizės procesui pažingsniui konstruojamas problemos sprendimo algoritmas.

3. Nuoseklius tikslinimus galima traktuoti kaip sąvokų hierarchijos konstravimą

- Problemos sprendimas aprašomas vartojant daugeliu būdų realizuojamos probleminės kalbos sąvokas;
- Žingsnis po žingsnio tos sąvokos yra išreiškiamos per paprastesnes ir galų gale yra pereinama prie kokios nors universaliosios programavimo kalbos sąvokų;
- Galima sakyti, kad šitaip mes pasirinkome tam tikrą probleminės kalbos realizavimo būdą ir realizavome tą kalbą pasirinktosios programavimo kalbos priemonėmis.

37. Paaiškinkite kas tai yra unifikavimo principas. (3)

Vieni programuotojai skaito kitų kodą, todėl kadavimas nuasmeninamas, t.y. nustatomas griežtas kodavimo standartas, vienodas visiems tos firmos programuotojamas.

- ***Unifikavimas*** – tai nebūtinų skirtumų (nevienodumu) pašalinimas. Svarbus PS inžinerijos principas, sakantis, kad PS turi būti projektuojama ir konstruojama, kaip **vienalytė**. Tam būtini standartai.

38. Paaiškinkite kas tai yra struktūrizavimo principas. (2)

Taisyklė: Visus artefaktus konstruok iš tipizuotų, patikimų elementų, kurių savybės yra gerai žinomos ir patikrintos. *Pvz.: D-struktūros.*

- Įrodyta, kad bet kuriai loginio pobūdžio problemos sprendimui aprašyti pakanka šių valdymo abstrakcijų: sekos, šakojimosi ir ciklo;
- Šią teoremą įrodė matematikai Bohm ir Jacopini (1966).

39. Paaiškinkite kas tai yra sistemų atvirumo principas. (2)

Taisyklė: Konstruok sistemas tik iš komponentų, tenkinančių atvirųjų sistemų standartus

- Šis principas riboja juodųjų dėžių naudojimą tam tikra šių dėžių (atvirųjų sistemų) klase
- Atvirųjų sistemų privalumai:
    - bet kurio komponento realizaciją galima keisti *(nekeičiant jo kontraktinius įsipareigojimus aprašančios specifikacijos)*;
    - vienoje sistemoje galima kombinuoti skirtingų gamintojų gaminamus komponentus.

40. Paaiškinkite kas tai yra interfeisų komfortiškumo principas. (2)

Interfeiso komfortiškumo principas: Visi PS vartotojo interfeisai turi būti komfortiški, t.y. jie turi būti pritaikyti konkretiems vartotojams (user-friendly), adaptyvūs, lengvai panaudojami (easy to use) kuo mažiau varginti vartotojus ir nesukelti jiems psichologinio diskomforto

- Sakoma, kad interfeisas yra pritaikytas konkretiems vartotojams (user-friendly), jei jis taip suprojektuotas, kad yra intuityviai suprantamas vidutiniam vartotojui ir aiškina pats save (self-explanatory).
- Sakoma, kad interfeisas yra lengvai panaudojamas (easy to use), jei juo naudojantis iš vartotojo nereikalaujama jokių bereikalingų veiksmų.

41. Paaiškinkite kas tai yra metaforizavimo principas. (3)

Metafora – žodžio reikšmės perkėlimas nuo vieno dalyko į kitą, siekiant antrąjį apibūdinti. Kartais vadinama paslėptu palyginimu be jungiamųjų žodelių, nurodančių gretinimo motyvus*(laivo nosis, gyvenimo saulėlydis)*. **Metafora***(PS kontekste)* – tai vieno objekto idėjos nusakymas kito, gerai žinomo objekto terminais. Metafora vieno objekto esmė nusakoma tapatinant ją su tam tikromis kito objekto savybėmis.

Taisyklė: Konstruok vartotojo interfeisą kaip dalykinės srities metaforą, sudarytą atsižvelgiant į vartotojų mentalitetą, tradiciją ir patirtį.

- Metaforos yra tam tikros abstrakcijos. Jos naudojamos kaip priemonė susieti tokius techninius ir sudėtingus dalykus kaip PĮ su vartotojo kasdieninio pasaulio realijomis
- Metaforos ypač naudingos, jei dalis ar visi vartotojo interfeise vaizduojamų objektų vartotojams yra nauji ir neįprasti *(pavyzdžiui, tokie dalykai, kaip failas, serveris)*
- Gera metafora padeda vartotojui susieti jam nežinomus objektus su žinomais
- Gerų metaforų pavyzdžiai: Darbo stalo metafora (MS Windows), įstaigos metafora (MS Office), e-pašto metafora (Pegasus Mail), lentelės metafora DB, formos metafora.
- Kartais projektuotojai teigia, kad jų suprojektuotame interfeise apskritai nėra metaforų. Tačiau šitaip būti negali. Kokia nors metafora visuomet panaudojama. Kitas dalykas, ar projektuotojas ją parinko sąmoningai, ar ne ir ar ji yra tinkamai parinkta. Problema ne tame, kad reikia naudoti metaforas, bet tame, kad jas reikia parinkti tinkamai ir kruopščiai suprojektuoti!

Rekomendacijos: 1.Įsitikink, kad metafora atitinka vartotojų prielaidas; ***2.***Vizualiai ji nebūtinai turi būti tiksli, kuo paprastesnė, tuo geriau. ***3.***Metaforų ieškok kasdieninėje vartotojo darbo aplinkoje; ***4.***Dažnai reikia kelių metaforų ***5.***Metafora apimk ne tik vaizdą, bet ir fukcionalumą ***6.***Suvok, kokia metafora naudojiesi, projektuodamas interfeisą ***7.***Nepersistenk – svarbu prisiminti, kad vartotojas nori matyti kaip skirtingus tuos objektus, kurie yra skiriami vykdant jo atliekamas užduotis, o ne tuos, kurie skirtingai realizuojami sistemoje! ***8.***Metaforos konstruojamos analizuojant vartotojo užduotis ir jo terminiją, o ne klasifikuojant PS objektus!

42. Paaiškinkite kas tai yra reaktyvumo principas. (6)

Agentinis ryšys sukuriamas tuomet, kai viena esybė įgalioja veikti kitą, jos vardu *(pvz.: verslo transakcijose)*.

Taisyklė: PS turi būti projektuojama kaip tikslingai veikiantis agentas. Tai reiškia, kad sistema privalo nuolat analizuoti dalykinės srities "reikalų būklę" ir tikslingai (arba bent jau iš anksto numatytu būdu) reaguoti į visas susidariusias "nenormalias" situacijas.

- Agentas suvokiamas kaip esybė, kuri, pasinaudodama savo sensoriais, geba stebėti savo aplinką ir kuri, panaudodama turimus efektorius, gali daryti tai aplinkai kokį nors poveikį. >>Russel and Norvigs
- Autonominis agentas – tai sistema, kuri yra situatizuota aplinkoje, yra tos aplinkos dalis, suvokia tą aplinką ir joje veikia. >>Franklin
- Mes domėmis sėkmingai veikiančiais rationaliais agentais (agentais "veikiančiais teisingai")
- Tikra *reaktyvioji PS* yra sistema, demonstruojanti refleksyvią ("sąlyga-veiksmas") elgseną

Pvz.: e-pašto sistema Pegasus Mail*(nuolat tikrina pašto dėžutę ir rodo kiek joje yra neskaitytų žinučių)*.

- Reaktyviojoje sistemoje iš anksto numatyta, kaip sistema turi reaguoti (kokių veiksmų imtis) į kiekvieną potencialiai galimą situaciją. Sistemoje nėra modeliuojama nei jos aplinka, nei PS, su kuriomis ji komunikuoja, todėl reaktyvioji sistema negali prognozuoti savo veiksmų pasekmių. Todėl ji neturi ir galimybių planuoti, kokį veiksmą geriau būtų atlikti. Reaktyvioji sistema neatlieka jokių išankstinių samprotavimų. Ji paprasčiausiai reaguoja į susidarančias situacijas iš anksto numatytu būdu.
    - *Proaktyvioji PS* - tai sistema, kuri realizuoja savo tikslingą elgseną perimdama iniciatyvą. Proaktyvioji sistema neleidžia susidaryti nepageidaujamoms situacijoms, iš anksto imdamasi atitinkamų priemonių.
    - *Planuojanti* PS - tai proaktyvioji sistema, kuri proaktyviąją elgseną realizuoja panaudodama "tikslų ir priemonių" samprotavimo metodą (means-end reasoning)
- Racionalius, tikslingus veiksmus atliekantis agentas turi naudotis išreikštiniu būdu užduotu simboliniu modeliu, modeliuojančiu: dalykinę sritį (aplinką), jo gebamus atlikti veiksmus, tikslus, kurių jam privalu siekti;
- nagrinėdamas potencialias galimybes (alternatyvius tikslus), sprendžia ką jam daryti ir įsipareigoja siekti pasirinktųjų tikslų; panaudodamas tikslų ir priemonių samprotavimo metodą (means-end reasoning), sudarinėja savo veiksmų planus (leistinų veiksmų grandines), kaip pasiekti pasirinktus tikslus ir galbūt tuos planus optimizuoja arba našumo, arba laukiamos naudos požiūriu.
- *Projektavimas, nenaudojant reaktyvumo principo* –projektuotojas iš anksto numato visas situacijas ir sistemos reakcijas į jas.
- *Projektavimas, naudojant reaktyvumo principa* – projektuotojas projektuoja tokią sistemą, kuri pati sprendžia, ką reikia padaryti, kad artėti prie numatytų tikslų.
- Agentinė PS inžinerija (APSI) – tai nauja PS kūrimo paradigma, susiformavusi objektinės PS inžinerijos (OPSI) pagrindu.
- APSI akcentuoja PS autonomiškumą, tarpusavio sąveiką, intelektuališkumą ir proaktyvumą.

Programų sistemų inžinerijos paradigmos

43. Populiariausios PSI paradigmos:

- Paradigma "iš viršaus žemyn" *(top-down approach)*
- Paradigma "iš apačios aukštyn" *(bottom-up approach)*
- Riešuto paradigma *(bootstrap approach)*
- Iteracinė paradigma *(prototyping)*
- Evoliucinio kūrimo paradigma *(incremental development)*
- Programų šeimų paradigma *(domain/application engineering)*
- Komponentinė paradigma *(reuse-based development)*
- Sintezės paradigma *(formal development)*

44. Paaiškinkite "iš viršaus žemyn" paradigmą. (2)

- **Pritaikyta** projektuoti komponentus, tenkinančius iš anksto užduotas specifikacijas.
- Taikant šią paradigmą pradedama nuo reikalavimų / vartotojo intefeiso projektavimo, nesigilinant į programos realizacija(nustatomos įeigos ir išeigos, bet pati PS traktuojama kaip juodoji dėžė). Intensyviai naudojami abstrakcijos principai dekomponuojant sistemą.
- Lipama laiptais žemyn iki kompiuterinės platformos lygmens, suformuojant kompiuterinės technologijos reikalavimus, garantuojančius interfeisų veikimą ir norimą sistemos elgseną.
- Projektavimo sprendimai priimami vadovaujantis funkciniais kriterijais:
    - problema dekomponuojama į komponentus (modulius, duomenų struktūras ar kt.),
    - komponentai realizuojami žemesnio lygmens (patikslintais) komponentais, ir artėja prie tikslinės platformos ypatumų.
- Apsaugo nuo kliento PS vizijos iškraipymo, tačiau analitikams iki galo neperpratus kliento poreikių, vizija vistiek gali tapti iškreipta.

45. Paaiškinkite "iš apačios aukštyn" paradigmą. (2)

- **Pritaikyta** linkusiems pirma įvertinti projektuosiamo komponento technologinį tinkamumą*(našumą ir pan.)* kuriamai PS, o tada projektuoti.
- Pirma pasirenkama kompiuterinė technologija, tada sprendžiama kaip jos priemonėmis kurti reikiamą PS. Intensyviai naudojamas abstrakcijos principas ir konkatenacijos operacijos.
- Viena iš populiariausių paradigmų, tinkanti nedidelėms PS kurti.
- Sudėtingesnių PS atveju dažnai iškraipoma kliento PS vizija, nes suprojektuojami interfeisai, kuriuos patogu realizuoti turimomis priemonėmis, o ne tokie, kurie būtų patogūs vartotojui.
- Projektavimo sprendimai priimami vadovaujantis technologiniais kriterijais:
    - komponentai kombinuojami, norint gauti aukštesnio lygmens komponentus, kurie, manoma, yra "tinkamesni" vartotojo interfeisams įgyvendinti

47. Paaiškinkite riešuto paradigmą. (2)

- PS kuriama kaip specializuota virtualioji mašina, turinti visas duomenų struktūras ir komandas, pritaikytas reikiamai problemai spręsti, o taip pat tą problemą sprendžiančią programą.
- Po to, žingsnis po žingsnio konstruojamos naujos, žemesnio lygmens virtualiosios mašinos, skirtos aukštesnio lygmens mašinoms įgyvendinti
- Kiekviena tokia mašina patikslina*(realizuoja)* aukštesnio lygmens virtualiosios mašinos komandas ir duomenų struktūras.
- Procesas baigiamas tada, kada visos aukštesnio lygmens mašinos komandos ir duomenų struktūros yra realizuotos kokia nors programavimo kalba
- Iš esmės, tai specialus paradigmos "iš viršaus žemyn" atvejis: Kiekvienas abstrakcijos lygmuo čia projektuojamas kaip virtualioji mašina. Pagrindinė idėja yra palaipsniui pertvarkyti probleminio pobūdžio virtualiąją mašiną į kompiuterinės platformos lygmens mašiną.
- Riešuto paradigmą galima traktuoti ir kaip specialų paradigmos "iš apačios aukštyn atveją" : šiuo atveju PS pradedama kurti nuo vadinamojo "branduolio", t.y. virtualiosios mašinos, realizuojančios bazines sistemos funkcijas; šios mašinos kalba programuojamos papildomos funkcijos, t.y. kuriama nauja virtualioji mašina, kuri tarsi "kevalas" padengia "branduolį"; procesas tęsiamas tol, kol sukuriama sistema, tenkinanti turimą reikalavimų specifikaciją.

48. Paaiškinkite iteracinę paradigmą.(2)

- kuriamas ir testuojamas būsimosios PS prototipas (*funkciniu ar kitais požiūriais neišbaigta PS)*;
- prototipas perdirbamas į labiau išbaigtą variantą*(naują prototipą)*;
- procesas tęsiamas kol galų gale sukuriama išbaigta PS, tenkinanti reikalavimų specifikaciją.

49. Paaiškinkite evoliucinę paradigmą. (2)

- Sistema yra kuriama kaip užsakovui vienas po kito pateikiamų papildinių *(increments)* seka:
  - suprojektavus visos sistemos architektūrą, kuriami ir pateikiami užsakovui tos sistemos "*branduolys*"*(svarbiausios funkcijos)* ir jo papildiniai;
  - kiekvienam papildiniui rengiama jo reikalavimų specifikacija ir jo projektinė dokumentacija;
  - kol kuriamas naujas papildinys, vartotojai dirba su einamąja PS versija ir išryškina jos trūkūmus.
- Evoliucinė paradigma yra panaši į iteracine, tačiau yra sukuriama geriau struktūrizuota PS ir jos kūrimo procesą yra lengviau valdyti:
  - PS funkcionalumas pateikiamas užsakovui anksčiau*(nors ir dalimis)*;
  - pradiniai papildiniai atlieka prototipų vaidmenį, tikslindami vėlesnių papildinių reikalavimus;
  - sumažėja projekto žlugimo rizika;
  - galima geriau išbandyti*(daugiau testavimo)* svarbiausias sistemos funkcijas.

50. Kas yra ekstremalusis programavimas? (2)

- Specialus*(šiuo metu labai populiarus)* evoliucinės paradigmos atvejis, kai kuriami ir pateikiami užsakovui labai maži papildiniai, paremtas nuolatiniu kodo tobulinimu, tiesioginiu klientu dalyvavimu PS kūrimo procese bei programavimu poromis; priskiriamas prie "agiliųjų metodų"
- termino "*ekstremalus*" dėl neigiamų asosiacijų pradedama atsisakyti.
- ***Programavimas poromis:*** prie ekrano sėdi dviese ir kartu rašo programą; šitaip vienas kitą tikrina ir išvengia daugelio klaidų; be to, kiekvieną programą žino mažiausiai du žmonės ir, jei vienam kas nors atsitiko, kitas gali tą programą keisti ir taisyti.

51. Paaiškinkite komponentinę paradigmą. (2)

- Sistema renkama iš turimų komponentų
  - grindžiama daugkartiniu rinkoje parduodamų komponentų panaudojimu;
  - didelę svarbą įgyja komponentų integravimo metodai;
  - labai svarbu, kiek reikia pastangų komponentui perprasti.
- PS kūrimas pradedamas komponentų analize ir jų reikalavimų keitimu.
- Dažnai komponentai surenkama tik dalis kuriamos PS, tad ši paradigma kombinuojama su kitomis.

52. Paaiškinkite programų šeimų paradigmą. (3)

- Apibrėžiama, apibrėžiant tam tikroje dalykinėje srityje jos sprendžiamų uždavinių *(problemų)* klasę
- Tai specialus komponentinės paradigmos atvejis, numatantis sprendžiamų problemų klasės nustatymą, ją sprendžiančių sistemų projektavimą, programų šeimos ruošinio kūrimą*(reikalavimai, kodas, projektinė/vartotojo dokumentacijos)*, bei konkrečių sistemų generavimą iš jo.
- Kadangi šeimai priklausančios programos turi daug bendrų savybių, prasminga jas apjungti, kad būtų galima pakartotinai panaudoti ne tik kodą, bet ir reikalavimus, architektūrą bei vartotojo dokumentacijos komponentus.
- Vadovaujantis šia paradigma, sistemos kūrimo procesas skyla į dvi dalis: 1. bendrybių paieška ir realizavimas (šeimos inžinerija; domain engineering) ir 2.konkrečių PS kūrimas (sistemos inžinerija; application engineering).
- Bendrybės*(commonality)* – tai savybės, kurias turi visos tai šeimai priklausančios PS
    - bendrybės tai ir yra tai, kas gali būti daug kartų panaudota pakartotinai (kiekvienai tai šeimai priklausančiai sistemai);
    - tuo pačiu bendrybės nustato šeimos ribas, nes per jas nusakoma tai šeimai priklausančių sistemų aibė.
- ***Šeimos inžinerija:*** Tam tikrai kokios nors dalykinės srities problemų klasei kuriamas programų šeimos ruošinys.
- ***Sistemos inžinerija:*** Iš ruošinio sukuriama konkreti tam tikrai dalykinei sričiai skirta PS.

53. Paaiškinkite karkaso paradigmą(FRAMEWORK'ai). (3)

- Tai specialus PS šeimos paradigmos atvejis, kai ruošinys kuriamas kaip objektinis ar kitas PS architektūrinis karkasas*(apibendrinta parametrizuota PS)*; konkreti PS sistema yra kuriama konkretizuojant karkasą*(parametrams skiriamos konkrečios reikšmes),* užpildant jį konkrečiu funkcionalumu; *pvz.: taip kuriamos vadinamosios verslo procesų valdymo sistemos (ERP paketai)*.
- ***Architektūrinis karkasas*** yra tam tikri PS „*griaučiai*": realizuojamos duomenų struktūros, modulių interfeisai, jų sąveika ir kiti konstrukciniai PS ypatumai; tačiau pats karkasas nėra užpildytas jokiu konkrečiu funkcionalumu t.y. jame pačių modulių realizacijos nėra.
- ***Objektinis karkasas*** kuriamas kaip susietų abstrakčių klasių rinkinys, karkase klasių realizacijos nėra:
    - abstrakčiosios klasės naudojamos paveldėjimo hierarchijoms konstruoti;
    - jos neturi realizacijų, jose apibrėžtos tik virtualios operacijos;
- Objektiniai karkasai iš esmės skiriasi nuo klasių bibliotekų:
    - bibliotekos tiražuoja programų kodą, karkasai realizuoja ir tiražuoja PS šeimai būdingas architektūrines bendrybes*(tipinius projektavimo sprendimus)*;
    - klasių bibliotekoje valdymo srautas vienkryptis – iš biblioteką naudojančios programos į bibliotekos klases;
    - objektiniame karkase valdymo srautas abipusis –programos klasės paveldi abstrakčių karkaso klasių ypatumus, bet virtualiosios operacijos, dinaminio susiejimo mechanizmu*(dynamic binding)*, susiejamos su jas realizuojančiu kodu programos vykdymo metu.
- Kuriant konkrečią PS šeimos sistemą, sukuriamas jo klasėmis numatytas funkcijų realizavimo kodas*(karkaso užpildas)*:
    - dažnai su karkasu yra tiražuojamas ir standartinis užpildas*(ar jo dalis)*;
    - Konkrečiai PS, standartinis užpildas kartais keičiamas nestandartine, tai PS būdinga realizacija.

54. Paaiškinkite sintezės paradigmą. (2)

- Formali sistemos specifikacija formaliomis taisyklėmis*(automatiškai)* perdaroma į veikiančią PS:
    - transformavimas atliekamas per kelis tarpinius žingsnius, jų metu neįneša klaidų;
    - todėl kodas visada atitinka specifikaciją ir atpuola būtinybė testuoti taip gautą kodą.
- ***Problemos:***
    - reikalingi specialūs įgūdžiai ir aukšta kvalifikacija;
    - sudėtinga formaliai specifikuoti kai kuriuos sistemos aspektus, pavyzdžiui, vartotojo interfeisą;
    - darant sistemos pakeitimus, prireikia sukurti naujas transformacijas ir įrodyti jų teisingumą;
    - kaip taisyklė, nepavyksta keisti šitaip kuriamų sistemų mastą (does not scale).
- ***Taikymų sritis:*** Kritinės sistemos.

Programų inžinerijos apibrėžimas

1. Dvi skirtingos termino programų inžinerija prasmės. (1)

Terminas programų inžinerija*(software engineering)* buvo įvestas, 1968 m. vykusioje, pirmojoje NATO konferencijoje PS inžinerijos klausimais. Nuo pat pradžios šis terminas turėjo 2 skirtingas prasmes:

- ***Siauroji prasmė:*** disciplinuotas principų, metodų ir įrankių naudojimas analizuojant kompiuterinių programų, jų eksploatavimo procedūrų ir jų dokumentacijos reikalavimus, visa tai projektuojant, realizuojant, eksploatuojant ir prižiūrint (McDermid, 1985)
- ***Plačioji prasmė:*** disciplina, užsiimanti PS kūrimo technikomis (Ramamoorthy at all, 1984)

Šios apibrėžtys yra visiškai skirtingos:

- 1-oje kalbama apie kompiuterines programas
- 2-oje kalbama apie PS, turint omenyje, kad PS sudaro įvairūs, skirtingos prigimties, komponentai*(failai, duomenų bazės, interfeisai, protokolai ir kt.)*, o programos yra tik vienas iš komponentų.

Šiame kurse PS inžinerija yra apibrėžiama šitaip:

- PS inžinerija – tai praktinė veiklos sritis, sistemų inžinerijos šaka, užsiimanti pramoninių PS kūrimu.
- PS inžinerija*(kaip mokslinė disciplina)* – tai inžinerinių mokslų šaka, nagrinėjanti pramoninių PS kūrimo principus, metodus, technikas, įrankius bei procedūras ir efektyvų jų panaudojimą PS pramonėje.

60. Kas vadinama PSI? Apibūdinkite dabartinę PSI būklę (4)

PSI yra disciplinuotas PS kūrimo būdas, apimantis:

- programų sistemos reikalavimų nustatymą ir dokumentavimą,
- fundamentinių projektavimo principų panaudojimą,
- projektinių alternatyvų analizę,
- vertinimus, ar galutinis produktas tikrai tenkina suformuluotus reikalavimus.

Esama padėtis programų/programų sistemų kūrime:

- dažniausiai amatas
- dažniausiai dirbama rinkai
- retkarčiais inžinerija

Kodėl PS inžinerija nėra stebuklinga burtų lazdelė? (3)

PSI turi savo problemų:

- Tai gana jauna inžinerijos šaka, per 42 metus*(1968-2010)* dar nepakankamai prigijusi praktikoje.
- Lyginant su kitomis inžinerijos sritimis, PS kūrimas vis dar yra netvarkingas, netgi chaotiškas.
- Tai vis dar daugiau amatas, o ne inžinerija, pirmiausia todėl, kad vis dar nemokama pakankamai tiksliai matuoti ir vertinti PS projektavimo kokybės.
- Kol kas dar niekas nežino geriausio PĮ kūrimo būdo.
- Tai, kas vadinama PS projektavimu, nėra išbaigtas projektavimas kitų inžinerinių disciplinų prasme; tai tik preliminarinis projektavimas, kuris turi būti detalizuojamas ir galbūt keičiamas kodavimo metu.

Sudėtinės PS inžinerijos dalys

27. Sudėtinės programų sistemų inžinerijos dalys: (4+1)

- Reikalavimų inžinerija
- Programų inžinerija
- Duomenų inžinerija
- Interfeisų inžinerija
- Web inžinerija
- Kokybės inžinerija:
  - *Patikimumo inžinerija (reliability)*
  - *Saugos inžinerija (safety)*
  - *Apsaugos inžinerija (security)*
  - *Našumo inžinerija (performance)*
  - *Panaudojamumo inžinerija (usability)*
  - *Prižiūrimumo inžinerija (maintainability)*
- PĮ priežiūra
- Įrankiai.

Kiekvienas iš PS inžinerijos dalių dar susideda iš:

- Teorinių pagrindų
- Inžinierinių pagrindų
- Architektūros
- Projekto valdymo
- Kokybės valdymo
- Konfigūracijos valdymo

3. Ką apima teoriniai PS inžinerijos pagrindai? Trumpai juos apibūdinkite. (3)

Teoriniai pagrindai: sąvokos, principai, paradigmos, modeliai, modeliavimo formalizmai, modeliavimo kalbos.

4. Ką apima inžineriniai PS inžinerijos pagrindai? Trumpai juos apibūdinkite.(3)

Inžineriniai pagrindai: tipiniai projektavimo sprendimai*(design patterns)*, karkasai *(frameworks)*, komponentai, įrankiai, technologijos

5. Ką nagrinėja disciplina „PS architektūros“? (3)

- architektūriniai stiliai;
- architektūrų savybės*(reikalavimai)*;
- modelinės*(reference)* architektūros:
  - įmonės sistemos architektūros*(enterprise architectures)*;
  - programų šeimų architektūros;
  - programų sistemų architektūros.
- jungimo mechanizmai*(glue mechanisms)*;
- įgyvendinimo būdai.

6. Ką nagrinėja disciplina PS projektų valdymas? (3)

Projekto valdymas:

- Projekto valdymo proceso komponentai*(subprocesai)*:
  - projekto iniciavimas,
  - projekto planavimas,
  - projekto vykdymas,
  - projekto kontrolė,
  - projekto užbaigimas.
- PS gyvavimo ciklo modeliai ir PS kūrimo procesai;
- Dalykinės sritys:
  - projekto integravimas,
  - terminų valdymas,
  - kokybės valdymas,
  - žmogiškųjų išteklių valdymas,
  - komunikavimo procesų valdymas,
  - rizikos veiksnių valdymas,
  - pirkimų valdymas.

Apskritai, kokybės valdymas yra projekto valdymo dalis (tokiu atveju parašysime ir apie kokybės valdymą). Kokybės valdymas – tai savarankiška PS inžinerijos šaka, nagrinėjanti: kokybės standartus, kokybės atributus, kokybės planavimą, kokybės kontrolę (vertinimą (evaluation), matavimą, testavimą ir pan.), kokybės užtikrinimo procedūras, kokybės įvertinimą (assessment).

7. Ką nagrinėja disciplina PS konfigūracijos valdymas? (3)

Konfigūracijos valdymas, kaip mokslinė disciplina, nagrinėja kaip užtikrinti, kad einamoji kuriamo produkto (ir jo komponentų) konfigūracija bet kuriuo laiko momentu būtų žinoma ir dokumentuota ir kad visi jos keitimai būtų kontroliuojami ir trasuojami, t.y.:

- pateikiamo produkto komponentų*(įskaitant jų versijas)* nustatymo ir identifikavimo metodus,
- tų komponentų atmainų*(release)* ir jų pokyčių kontrolės visose gyvavimo ciklo stadijose metodus,
- einamosios produkto komponentų būklės ir reikalavimų ją keisti dokumentavimo metodus,
- produkto komponentų išsamumo ir korektiškumo tikrinimo procedūras.

8. Ką nagrinėja disciplina PS reikalavimų inžinerija? (3)

Reikalavimų nustatymas, reikalavimų dokumentavimas, reikalavimų priežiūra, reikalavimų modeliavimas, reikalavimų analizė ir vertinimas, reikalavimų lokalizavimas, reikalavimų operacionalizavimas ir nuleidimas žemyn.

9. Ką nagrinėja programų inžinerija?(3)

Programų specifikavimas, programų projektavimas, programų moduliarizavimas, programų kodavimas ir kodo dokumentavimas, programų testavimas ir derinimas.

10. Ką nagrinėja duomenų inžinerija? (3)

Duomenų inžinerija, kaip mokslinė disciplina, nagrinėja, kaip PS projektuoti, realizuoti, tvarkyti, prižiūrėti ir naudoti duomenų struktūras ir pačius duomenis, t.y.:

- duomenų bazių projektavimą;
- metaduomenis ir jų vaizdavimo bei apdorojimo metodus;
- kalbas duomenims, prieigai prie jų ir manipuliavimu jais aprašyti;
- prieigos prie duomenų, jų apsaugos ir darnos (integrity) kontrolės strategijasir mechanizmus.

11. Ką nagrinėja interfeisų inžinerija? (3)

Pagrindinis šios disciplinos nagrinėjimų objektas yra vartotojo interfeisų (bet nagrinėjami ir kitokie interfeisai) specifikavimas, projektavimas ir realizavimas. Vienas iš pagrindinių tikslų yra didinti vartotojo darbo našumą. Nors mes priskiriame šią discipliną PS inžinerijai, ji taip pat yra priskiriama žmogaus ir kompiuterių sąveiką nagrinėjančiai mokslo šakai. Interfeisų inžinerija nagrinėja

- interfeisų architektūras, tipinius interfeisų projektavimo sprendimus (design patterns), interfeisų standartus, įrankius interfeisams kurti, interfeisų prototipų kūrimo technikas.

12. Ką nagrinėja Web inžinerija?(3)

Web inžinerijos nagrinėjimų objektas yra Web sistemų specifikavimas, projektavimas, realizavimas, aptarnavimas ir priežiūra. Ji nagrinėja:

- informacijos architektūras, informacinės erdvės struktūrizavimo ir dekomponavimo į hipermedia puslapius technikas, hipermedia puslapių dekomponavimo į smulkesnius daugkartinio panaudojimo objektus technikas, vizualizavimo technikas, Web sistemų kūrimo įrankius.

13. Ką nagrinėja kokybės inžinerija? (3)

Kokybės inžinerija nagrinėja inžinerines veiklas, naudojamas užtikrinti reikalaujamą kuriamo produkto kokybę, t.y.: *patikimumo,saugos, apsaugos, našumo, panaudojamumo ir prižiūrimumo inžinerijas*.

14. Ką nagrinėja patikimumo inžinerija? (3)

Ši disciplina numato, kokios vadybinės ir inžinerinės procedūros turi būti naudojamos projekte, siekiant užtikrinti pakankamą dėmesį visoms projekto detalėms, vienaip ar kitaip darančioms įtaką kuriamos PS patikimumui. Ji nagrinėja:

- patikimumo modelius,
- patikimumo reikalavimus ir standartus,
- patikimumo matavimo ir vertinimo metodus,
- patikimumą užtikrinančio projektavimo būdus*(designing for reliability)*,
- patikimumo testavimo metodus ir procedūras,
- testų patikimumui testuoti projektavimo metodus.

15. Ką nagrinėja našumo inžinerija? (3)

Našumo *(performance)* inžinerija kaip visose gyvavimo ciklo stadijose planuoti ir įgyvendinti reikalaujamą PS našumą, t.y.:

- našumo modelius,
- našumo reikalavimus ir standartus,
- našumo matavimo ir vertinimo metodus,
- našumą užtikrinančio projektavimo būdus (performance-oriented design)

16. Ką nagrinėja apsaugos inžinerija? (3)

- PS apsaugos reikalavimai, juos pažeidžiančių grėsmių nustatymas ir pažeidimų pasėkmių likvidavimo strategijos,
- projektavimo ir programavimo metodai, padedantys išvengti sąmoningų ir atsitiktinių PS apsaugos pažeidimų,
- saugūs PS išskleidimo (deployment) metodai,
- PS apsauga eksploatavimo metu.

17. Ką nagrinėja saugos inžinerija? (3)

Saugos inžinerija nagrinėja:

- sistemos saugą pažeidžiančių rizikos veiksnių (hazards) analizės metodus,

(sistemos saugos rizikos veiksniais vadinamos sistemos būsenos, galinčios sukelti nelaimingus atsitikimus).

- saugiai naudojamo ir saugaus programinio produkto ribojimų nustatymo metodus,
- saugos reikalavimų specifikavimo ir analizės metodus,
- saugą užtikrinančio projektavimo (design for safety) metodus,
- saugos požiūriu kritinių sistemų testavimo ir vertinimo metodus,
- saugos standartus ir PS saugos sertifikavimo metodus,
- veikiančių PS saugos monitoringo metodus.

18. Ką nagrinėja panaudojamumo inžinerija?(3)

Panaudojamumo (usability) inžinerija, atsižvelgdama į žmogaus ir kompiuterio sąveikos ypatumus, nagrinėja kaip suprojektuoti tokią PĮ, kuri priimtų dėmesin žmogaus psichofiziologines savybes ir kuria žmogui būtų patogu ir lengva naudotis. Iš esmės, tai yra ergonomikos šaka. Ji glaudžiai siejasi ir su interfeisų inžinerija. Disciplina nagrinėja:

- panaudojamumo lygmenis, panaudojamumo standartus, panaudojamumo gerinimo būdus, metodus, technikas ir tam skirtas veiklas, panaudojamumo matavimo ir vertinimo metodus.

19. Ką nagrinėja disciplina PS prižiūrimumo inžinerija?(3)

Prižiūrimumo inžinerija nagrinėja, kaip, kuriant PS, palengvinti tų sistemų priežiūrą ir sumažinti priežiūros kaštus:

- prižiūrimumo standartus;
- prižiūrimumo gerinimo būdus, metodus, technikas ir tam skirtas veiklas;
- lenviau prižiūrimų sistemų projektavimo metodus (design for maintainability);
- prižiūrimumo vertinimo ir matavimo metodus.

20. Ką nagrinėja instrumentinių sistemų teorija?(3)

Instrumentinių sistemų teorija nagrinėja programinių įrankių kūrimo metodus, t.y.: programavimo ir PS kūrimo aplinkų architektūras, įrankių realizavimo ir integravimo metodus, repozitorijus, įrankių sąveikos standartus.

Chroniška programavimo krizė

Krizė kyla dėl šių didelių PS savybių:

- Neužtikrina reikiamo funkcionalumo
- Yra kuriamos per ilgai
- Reikalauja per daug lėšų joms sukurti
- Vykdymui reikalauja per daug resursų (laiko, atminties)
- Yra nepatikimos
- Negali evoliucionuoti ir prisitaikyti prie kintančių vartotojų poreikių

Chroniška programavimo krizė: funkcionalumo aspektas. (4)

Užsakovams pateiktos IS, PĮ funkcionalumas iš tiesų neatitika to, ką apie jį tvirtino vykdytojai. To priežastis yra atotrūkis:

- tarp verslo vizijos ir verslo procesų
- tarp verslo procesų ir technologinės infrastruktūros
- tarp tecnologijos naudojimo reikalavimų ir darbuotojų kompetencijos
- tarp strateginių ir taktinių veiksmų

Kad problemos nebūtų, reikia:

- susieti strategiją ir tikslus su vizija;
- projektuoti tokius verslo procesus, kurie padeda pasiekti norimus tikslus ir įgyvendinti verslo viziją;
- taip pertvarkyti organizacijos struktūrą, kad ji būtų suderinta su verslo procesais, strategija ir vizija;
- darbuotojų kompetencijos reikalavimus formuluoti atsižvelgiant į verslo procesų ir organizacinės struktūros ypatumus;
- technologinę infrastruktūrą projektuoti atsižvelgiant tiek į darbuotojų kompetenciją, tiek ir į organizacijos viziją, misiją, strategijas, tikslus, verslo procesus ir struktūrą;
- veikimo taktiką planuoti tik po to, kai verslo procesai, organizacijos struktūra, darbuotojų kompetencijos reikalavimai ir technologinė infrastruktūra yra suprojektuoti iki galo.

Chroniška programavimo krizė: kainos aspektas. (6)

- aparatūros kaina drastiškai sumažėjo, tuo tarpu PĮ tokiu pačiu greičiu didėja.
- programinės įrangos kaina taip pat mažėjo, tačiau augo kuriamos programinės įrangos apimtys
- Produkto kaina nusistovėjo, tačiau naujų versijų kurimas užima vis daugiau resursų.
- Išlaidos PĮ auga apie 12% per metus.
- JAV programinė įranga sudarė:
    - 1980 m. 2% bendrojo nacionalinio produkto
    - 1985 m. 8 % bendrojo nacionalinio produkto
    - 1990 m. 13 % bendrojo nacionalinio produkto
    - 1998 m. 38 % bendrojo nacionalinio produkto
- Net ir nedidelis darbo našumo augimas pasauliniu mastu ženkliai sumažintų išlaidas programinei įrangai!
- JAV atlikti tyrimai(1995): dėl žlugusių PĮ kūrimo projektų per metus prarandama apie 81 milijardą JAV dolerių
- Programos kodo eilutės kaina svyruoja nuo 100 iki 1000 JAV dolerių
- Projektas dažnai užsitesia, viršija pradinį biudžetą virš 50%.
- Augant PĮ kūrimo projekto dydžiui, išlaidos vienam funkciniam taškui taip pat didėja. (Užduotys sudėtingėja, tad jom įgyvendinti reik vis daugiau žmonių).
- Dideliuose PĮ kūrimo projektuose taip pat ženkliai išauga ir pridėtinės išlaidos.

Kodėl PS yra tokios brangios? (5)

- Didėjant projektui, jis darosi sudėtingesnis
- Didėjant projektui, didėja užduočių skaičius
- Didėjant projektui, didėja vykdytojų skaičius, o tuo pačiu ir išlaidos vadybai.
- Kadangi PĮ kūrimo projektuose fiksuotos išlaidos yra minimalios, projektų apimties didinimas dažniausiai ekonomiškai neatsiperka.
- Programas nesunku pavogti, todėl gamintojai priversti mažinti jų kainas
- Didelėse progamose klaidų taisymas labai brangus.

Chroniška programavimo krizė: ataskaita apie chaosą. (4)

1995m - 8380 projektų ataskaita:

- 53% projektų vėluoja bei viršija pradinį biudžetą;
- 16% sėkmingi;
- 31% nebaigiami;

2000m - 280000 projektų ataskaita:

- 49% projektų vėluoja bei viršija pradinį biudžetą;
- 28% sėkmingi;
- 23% nebaigiami;

1979 m. JAV Statistikos departamento studija atskleidė, kad pagal užsakymus kurtai PĮ:

- 2 % sukurta PĮ buvo naudojama be pakeitimų
- 3 % sukurta PĮ buvo naudojama ją šiek tiek perdarius
- 45 % pradėjus naudotis sukurta PĮ, jos buvo arba atsisakyta, arba ji buvo iš esmės perdaryta
- 20 % PĮ buvo sukurta, bet ja nepradėta naudotis
- 30 % PĮ apskritai nebuvo sukurta, nors vykdytojams už ją buvo sumokėta
- Priežiūros kaštai viršija 60% visų programų sistemos kūrimo kaštų
    - 20% klaidų taisymas
    - 30% keitimai, susiję su reikalavimų ir/arba TĮ pokyčiais
    - 50% tobulinimas (papildomas funkcionalumas)
- Klaidų šalinimas projektavimo stadijoje gali kainuoti nuo 10 iki 100 kartų mažiau, nei klaidų šalinimas programavimo metu.

Chroniška programavimo krizė: baigimo terminai. (6)

- 2002 m. atlikta IT organizacijų apžvalga nustatė, kad:
    - 78% organizacijų teisėsi teismuose su užsakovais;
        - 56% iš jų dėl to, kad projekto baigimas nukelta net kelis kartus;
- Netikslūs projekto kainos ir trukmės vertinimai sužlugdo daugelį projektų. Tenka spausti terminus, kad spėti pagal biudžetą.
- Pradėjus vėluoti bandoma mažinti funkcionalumą, testavimą. (kas savo ruožtu gali dar labiau prailginti darbo laiką.)

Vėluojama, nes:

- blogas planavimas
- neatliekamas reikalavimų valdymas
- neatliekamas pokyčių valdymas
- neatliekamas konfigūracijos valdymas
- per mažas darbo našumas
- prastas darbų koordinavimas
- prastas projekto valdymas
- vykdytojai per daug laiko sugaišta svarstymams ir aptarimams

Chroniška programavimo krizė: patikimumo aspektas. (6)

- 2002 m. atlikta IT organizacijų apžvalga nustatė, kad
  - 78% organizacijų teisėsi teismuose su užsakovais
    - 45% dėl to, kad dėl klaidų nebuvo galima naudotis programom.
- Apie pusę naudojamų programų turi rimtų klaidų.
- Apie 90% trykių vyksta dėl ne daugiau kaip 10% programose esančių klaidų.
- Patikimumas žemas, nes:
  - blogas projektavimas
  - technologinės disciplinos stoka
  - nėra kokybės planavimo, kontrolės ir užtikrinimo
- Patikimumo didinimas:
  - Kolegų peržiūros (peer reviews) pašalina apie 60% klaidų
  - Griežtai reglamentuotas programuotojo asmeninio darbo technologinis procesas gali sumažinti klaidų skaičių net 75%

Kaip suvaldyti chroniską programavimo krizę? Priklausomybių modelis. (4)

- Atsisakyti ikipramoninių darbo metodų.
- Atotrūkių pašalinimas:
  - Vizija ir siekiai
  - Strategijos ir tikslai
  - Verslo procesai
  - Organizacinė struktūra
  - Darbuotojų gebėjimai
  - Technologinė infrastruktūra
  - Taktika

Paskaitų klausimai-atsakymai

3 paskaita

1. Ar galima programas traktuoti kaip meno kūrinius? Kuo programų sistemų inžinerija skiriasi nuo meno? (2)

Programas galima traktuoti kaip meno kūrinius, jei programuojama siekiant estetinio vaizdo, originalumo, nėra griežtų kodavimų standartų, nesiekiama, kad programas galima būtų tiražuoti dideliais tiražais, lengvai perdirbti jas į naujas programas.

PSI skiriasi nuo meno:

- Kodas neturi estetinio vaizdo
- Meninę vertę gali suprasti nebent specialistas
- Apibrėžtumu, dokumentacija

2. Amato ir meno panašumai bei skirtumai. Programinės įrangos kūrimas kaip amatas. Kuo programų sistemų inžinerija skiriasi nuo amato? (2)

- ***Panašumai:*** Abiems reikalingas kūrybiškumas ir įgūdžiai, o taip pat tam tikros žinios. Abiejų mokomasi panašiai.
- ***Skirtumai:*** Amatas siejamas su praktinį panaudojimą turinčių daiktų darymu. Gaminama pagal užsakymą, lieka mažiau vietos kūrybinei veiklai.
- ***Programų kūrimas kaip amatas:*** Mažos įmonės užsiima programavimu kaip amatu. Tiesioginis bendravimas, supaprastinti sistemos darymo žingsniai, sistemos architektūra slepiasi kode *(o ne dokumentuose)*.
- ***PSI skirtumai nuo amato:*** skiriasi sprendimų priėmimas*(pagal griežtas taisykles)*, skiriasi mokymasis.

3. Ar galima programinės įrangos tyrinėjimus traktuoti kaip mokslą? Ar galima programų sistemų inžineriją traktuoti kaip mokslo šaką? (2)

PSI galima traktuoti kaip mokslo šaką, bet tik iš dalies. PSI labiau yra orientuota į mokslo pasiekimų praktinį pritaikymą ir tiražavimą, o ne į tyrinėjimus.

4. Inžinerijos ir mokslo panašumai bei skirtumai. Svarbiausi amato ir inžinerijos skirtumai. Kokios inžinerinės disciplinos naudojamos visose inžinerijos šakose? (3)

Inžinerijos ir mokslo panašumai: gabūs mokytis profesionalai, teorijos naudojimas rezultatams pasiekti.

Inžinerijos ir mokslo skirtumai: mokslas griežtesnis už inžineriją. Jame reikia ne tik pademonstruoti kad veikia, bet ir paaiškinti, kodėl taip veikia. Inžinerijoje rezultatai tiesiogiai siejami su ekonomika, naudojamasi ne tik teorija bet ir analize, remiamasi racionalumu.

Svarbiausi amato ir inžinerijos skirtumai: Inžinerijos tikslumas ir smulkmeniškumas gamyboje, darbo metodai (formalūs ir griežtai aprašyti vs neformalūs ir intuityvūs). Daug griozdiškos dokumentacijos :)

Inžinerinė disciplina, naudojama visose inžinerijos šakose: Sistemų inžinerija. Tai labai bendra disciplina. Ji nagrinėja bendruosius principus ir metodus, pritaikomus kuriant bet kurio pobūdžio sistemas, įskaitant verslo sistemas, informacijos sistemas ir programų sistemas.

7. Kuo skiriasi sistemos probleminė ir dalykinė sritys? (1)

Dalykine sritimi nusakome sistemos panaudos sritį(pvz. matuoti „*kažką*“), o probleminė sritimi nusakome kokia yra mūsų sistemos paskirtis(pvz. „*tiksliai matuoti laiką*“)

8. Sistemų inžinerijos kaip technologinio proceso traktuotė. (5)

Technologinis SI procesas: Poreikiai -> Problema -> Alternatyvų analizė -> Sistemos modeliavimas -> Integravimas -> Diegimas -> Našumo matavimai -> Procesas ir produktas.
Kiekvieną proceso žingsnį galima vertinti ir prireikus keisti*(iki tobulybės ar kol atsibos)*.

Problemos formulavimas pradedamas, aprašant arba planuojamas sistemos aukščiausiojo lygmens funkcijas, arba tas jau esamos sistemos savybes, kurias reikia pagerinti

Alternatyvūs sistemos reikalavimų įgyvendinimo būdai*(sistemos projektai)* vertinami našumo ir kainos kriterijais. Dažnai nei vienas būdas nėra geriausias. Viskas priklauso nuo vertinimo kriterijų prioritetų. Būdai *modeliuojami*. Pasirinktos alternatyvos modelis – detalizuojamas, juo naudojamasi visą tolimesnį kūrimo laikotarpį.

Integravimas – tai komponentų jungimas į vieną veikiančią visumą, kuriančią reikiamus rezultatus. Jo metu taip pat yra konstruojami tiek išoriniai pačios sistemos, tiek ir vidiniai jos posistemių interfeisai.

Sistemos diegimas – tai pradėjimas naudoti tiems tikslams, kuriems ji ir buvo kurta. Jis apima pradinius sistemos paleidimo darbus ir pradėjimą gaminti planuotą sistemos išeigą. Sistemos naudojimo ir priežiūros metu turi būti nuolat matuojamas jos *našumas*

Pakartotinas vertinimas atliekamas stebint sistemos išeigą ir panaudojant surinktą informaciją darbui tobulinti, keičiant sistemos įeigą arba darbo su ja procesą. Tai - vienas iš svarbiausiųjų inžinerijos metodų.

9. Funkcinės ir fizinė sistemos dekompozicijos skirtumai. (3)

Vienas fizinis komponentas gali vykdyti keletą funkcijų, bei atvirkščiai - vieną funkciją gali vykdyti keli fiziniai komponentai.

Viena iš sistemų inžinieriaus užduočių yra atlikti kuriamos sistemos funkcinę dekompoziciją: **(1)** susieti sistemos funkcijas su jos (fiziniais) komponentais ir įsitikinti, kad už kiekvienos funkcijos vykdymą yra atsakingas konkretus komponentas *(arba keli komponentai)* **(2)** susieti sistemos funkcijas su jos reikalavimais **(3)** įsitikinti, kad sistema vykdys visas reikalingas užduotis ir nevykdys jokių nereikalingų užduočių.

Pvz: Funkcinė ir fizinė dekompozicija:
- *Funkcija - pakilimas ir nusileidimas. Fizinis komponentas - kojos.*
- *Funkcija - vietos ir greičio pojūtis. Fizinis komponentas - akys.*
- *Funkcija - navigavimas. Fizinis komponentas - smegenys.*
- *Funkcija - horizontalus judėjimas. Fizinis komponentas - sparnai.*
- *Funkcija - vertikalus judėjimas. Fizinis komponentas - sparnai.*

12. Kokie yra sistemos įgyvendinamumo analizės tikslai? (2)

Išsiaiškinti ar sistemą įmanoma sukurti bei ar konkrečiai organizacijai yra verta ją kurti.

16. Kas vadinama apibendrinta verslo sistema? (2)

- ***Tyrimai/konstravimas*** – technologija, proceso ir produkto projektavimas, patentai
- ***Gamyba*** – technologija, žaliavos, galingumai, patalpos, pirkimai
- ***Marketingas*** – kainodara, reklama, pakavimas, prekės ženklai
- ***Pardavimas***, ***platinimas*** – pardavėjai, kanalai, prekių sąrašas, sandėliavimas, transportavimas
- ***Paslaugos*** – garantijos, greitis, pateikimas, kainos

16. Kokie yra verslo sistemos lygmenys? (1)

1 - kokybės*(metodų)* vadovas, **2** - procedūros, **3** - pareigybinės instrukcijos, **4** - išoriniai dokumentai.

17. Kokie yra verslo proceso elementai? (3)

Verslo procesai turi įeigas ir išeigas. Proceso rezultatus visada galima identifikuoti ir skaičiuoti. Visi verslo procesai yra siejami su užsakovais ir jų poreikiais. Užsakovo požiūriu, proceso resultatai yra vienintelė priežastis, kodėl tas procesas apskritai egzistuoja.

18. Kaip tarpusavyje susijusi verslo inžinerija ir sistemų inžinerija? (2)

Verslo inžinerija iš sistemų inžinerijos paveldi visas jos sąvokas ir visus jos metodus, tačiau sąvokos ir metodai yra suvokiami, tikslinami, konkretinami ir išplečiami, atsižvelgiant į konkrečios sistemų klasės, verslo sistemų, ypatumus.

19. Kokie yra VPR principai? (2)

- Organizuok darbą pradėdamas nuo išeigos;
- Tobulink technologijas;
- Nustatyk su procesu susijusias **problemas***(pvz.: brangu/ilgai trunka)* ir jų **priežastis***(pvz.: skaičiavimai ranka)*

20. Kokie yra IS komponentai? (2)

IS komponentai:

- ***Žmonės*** – vartotojai, vadybininkai, IS kūrėjai
- ***Duomenys*** – medžiaga informacijai kurti
- ***Technologija*** – techninė ir programinė įranga, palaikanti kitus IS komponentus

21. Kuo skiriasi duomenys ir informacija? (1)

Duomenys yra faktai apie įmonę ir jos atliekamas verslo transakcijas. Patys savaime duomenys yra mažai prasmingi ir naudingi. Informacija įprasmina duomenis. Duomenys įprasminami juos interpretuojant, tikslinant ir apdorojant.

22. Kokios yra IS inžinerijos stadijos? (1)

- ***Planavimas*** – verslo problemos nustatymas, jos ribų nustatymas, jos sprendimo tikslų ir sprendimo strategijos planavimas
- ***Analizė*** – reikalavimų, kuriuos turi tenkinti verslo problemos sprendinys, formulavimas ir analizė.
- ***Projektavimas*** – problemą sprendžiančios sistemos projektavimas.
- ***Realizavimas*** – sistemos sukonstravimas.
- ***Priežiūra****(palaikymas)* – įgyvendinto sprendinio analizė, tikslinimas ir tobulinimas.

22. Kaip tarpusavyje susiję informacinių sistemų inžinerija(ISI) ir bendroji sistemų inžinerija(SI)? (1)

IS inžinerija yra bendrosios SI konkretizacija. Ji paveldi visas SI sąvokas ir metodus, tačiau jie yra suvokiami, tikslinami, konkretinami ir išplečiami, atsižvelgiant į konkrečios sistemų klasės - IS - ypatumus.

26. Programinio produkto pobūdis. (5)

- Programinė įranga nematoma, jos negalima pačiupinėti. Programos struktūra paslėpta, jos elgsena - vienintelė pastebima išorinė apraiška. Programas sunku aprašyti ir vertinti.
- Programinės įrangos kūrimo pramonė reikalauja didelių darbo sąnaudų. Programuotojo darbą sunku automatizuoti.
- Programinė įranga nėra gaminama; ji projektuojama. Gamyba (kopijavimas) yra labai paprasta, kaina ir sudėtingumas slypi programinės įrangos kūrime.
- Tariamas darbo paprastumas: net ir neapmokyti žmonės gali sukurti ką nors veikiančio - trūkumai išryškėja gerokai vėliau. Nesunku keisti - tačiau žmonės gali ne iki galo suprasti kokias pasekmes turės pakeitimas.
- Programinė įranga veikdama “nesusidėvi”. Ji degeneruoja darant pakeitimus.
- Programinės įrangos projektavimas yra neįtikėtinai brangus. Taip yra todėl, kad programinė įranga yra ypatingai sudėtingas gaminys, ir todėl, kad praktiškai beveik visi programinės įrangos kūrimo projekto etapai traktuotini kaip projektavimo proceso žingsniai.
- Sudėtingumo kaina - programų sistema dažnai per sudėtinga, kad ją aprėptų ir perprastų vienas žmogus. Programas sunku aptarinėti.

27. Kodėl programinė įranga yra tokia sudėtinga? (2)

PS sprendžia labai sudėtingus uždavinius. Jų kūrimo procesą sunku valdyti. Net ir mažos klaidos programose gali sukelti dideles pasekmes. PĮ privalo aptarnauti sudėtingus išorinius interfeisus, veikti su realaus pasaulio fizinėmis sistemomis. Nuolatiniai PĮ keitimai - užsakovai keičia reikalavimus ir nori vis naujo funkcionalumo, tech. įranga nuolat kinta. Pakeitimai daromi, net ir nesikeičiant funkcionalumo reikalavimams.

43. Apibūdinkite svarbiausius PĮ raidos etapus. (3)

1 etapas – paketinis darbo režimas, ribotas platinimas, užsakomoji PĮ
2 etapas – realaus laiko darbo režimas, laiko skirstymo sistemos, duomenų bazės, visuotinio naudojimo PĮ
3 etapas – išskirstytos sistemos, įmontuotas intelektas, pigi TĮ, dėmesys vartotojui
4 etapas – galingos darbo stotys, objektinės technologijos, ekspertinės sistemos, dirbtiniai neuroniniai tinklai, lygiagretieji skaičiavimai

44. Apibūdinkite svarbiausius TĮ raidos etapus. (3)

1 etapas - pikas 1945-1955 m. – tyrimams skirtos darbo stotys *(mainframes)*
2 etapas - pikas 1955-1966 m. – komercinės darbo stotys
3 etapas - pikas 1966-1977 m. – komercijai skirti mini-kompiuteriai
4 etapas - pikas 1977-2000 m. – asmeniniai kompiuteriai
5 etapas - pikas 2000- ... m. - internetas, visuotiniai skaičiavimai *(lygiagretūs, išskirstyti, ir t.t.)*

57. Ką apima programos kūrimas? (1)

Programos kūrimas apima naujų programų kūrimą, esamų programų modifikavimą, jų pakartotiną panaudojimą, reinžineriją, priežiūrą ir bet kitus darbus, pasibaigiančius programinio produkto sukūrimu.

58. Ar „rašydami" programą jūs ją projektuojate? Kuo skiriasi PS projektavimas ir programos kodavimas? (2)

Rašant programą, mintyse ji tam tikrame lygyje projektuojama – apmąstomos loginės sąsajos kode ir t.t. PS projektavimas yra programos planavimas ir dokumentavimas, kodavimas yra kodo rašymas pagal ją.

59. Kokie yra svarbiausieji PS inžinerijos ir programavimo skirtumai? (1)

- ***Programavimas*** - dirba individai, kuriamos programos, trumpas gyvavimas, rašoma nuo pradžios, minimali priežiūros kaina.
- ***PS inžinerija*** - dirba kolektyvai, kuriamos sudėtingos PS, ilgas gyvavimas, naudojami jau paruošti komponentai, priežiūrai skiriama daugiausia pinigų.

61. Kuo skiriasi PSI, kompiuterių inžinerija ir teorinė informatika? (3)

Lyginant su kompiuterių inžinerija, labiau domisi PĮ, o ne TĮ ir koncentruojasi į didesnes taikomojo pobūdžio programas. ***Lyginant su teorine informatika***, tai labiau praktikai pritaikytas mokslas, akcentuojantis visą PĮ kūrimo procesą, nuo pradinėsį iki galutinio produkto. Griežtesnė disciplina nei teorinė informatika, naudoja labiau susistemintas praktikas, padedančias užtikrinti kuriamų produktų patikimumą ir saugumą.

62. Kuo skiriasi PS inžineriaus, tradicinio inžinieriaus ir teorine informatika užsiiminėjančio mokslininko darbas? (2)

Mokslininkas*(dirbantis teorinės informatikos srityje):*

- Kuria algoritmus ir įrodinėja jų teoremas; projektuoja programavimo, modeliavimo bei specifikavimo kalbas, kuria žinių vaizdavimo schemas, …
- Dirba, neturėdamas griežtai nustatytų terminų

Tradicinis inžinierius:

- Užsakovo pavedimu, sprendžia specifines savo dalykinės srities *(pvz., statybos)* inžinerines problemas
- Yra kompiuterių, projektavimo kalbų, įrankių, technikų ir metodų vartotojas

Programų sistemų inžinerius:

- Dirba daugelyje dalykinių sričių
- Dirba pagal griežtai nustatytus terminus

63. Kaip yra tarpusavyje susiję programų sistemų inžinerija ir sistemų inžinerija? (2)

PS inžinerija – tai bendrosios sistemų inžinerijos konkretizacija. Ji numato, kaip panaudoti bendruosius sistemų inžinerijos metodus, kuriant programų sistemas. PSI iš sistemų inžinerijos paveldi visas jos sąvokas ir visus jos metodus, tačiau sąvokos ir metodai yra suvokiami, tikslinami, konkretinami ir išplečiami, atsižvelgiant į konkrečios sistemų klasės, programų sistemų, ypatumus.

64. Kokie yra svarbiausieji IS inžinerijos ir PS inžinerijos skirtumai?

ISI užsiiminėja tik informacijos apdorojimo sistemomis, PSI nagrinėjamų sistemų klasė šia prasme platesnė.

PSI užsiiminėja tik programine įranga, todėl, kalbant apie informacijos apdorojimo sistemas, domimasi tik kompiuterizuotomis IS, tiksliau, tik tokių sistemų PĮ, paliekant nuošalyje kitus klausimus.

- Šia prasme ISI nagrinėjamų klausimų ratas yra platesnis, nes jis apima ir visus rankinio informacijos apdorojimo aspektus
- ISI verslo, kultūros, socialiniai ir organizaciniai klausimai yra ne mažiau svarbūs už techninius klausimus

65. Kaip yra tarpusavyje susietos įmonės sistema, verslo sistema, informacinė sistema ir programų sistema? (2)

Įmonės sistema apima bent vieną verslo sistemą, ją palaikančias informacines sistemas ir tų informacinių sistemų programinę įrangą.

66. Kaip siejasi tarpusavyje SI, verslo inžinerija, IS inžinerija ir PS inžinerija? (2)

Verslo, informacinių sistemų ir programų sistemų inžinerijos yra sistemų inžinerijos rūšys, konkretizuojančios ir išplečiančios bendrus sistemų inžinerijos sąvokas ir metodus, atsižvelgiant į konkrečios inžinerijos šakos ypatumus.

Verslo, informacinių sistemų ir programų sistemų inžinerija palaiko ir leidžia pradėti sistemos kūrimo projektą.

4 paskaita

Pramoninės ir amatinės gamybos skirtumai? (3)

Pramoninė gamyba	*Amatas*
Aukštas darbų mechanizavimo ir automatizavimo lygis	Rankinis darbas, labai specializuota įranga
Masinė gamyba	Vienetinė gamyba
Gaminių standartizavimas	Gaminių individualizavimas
Grupinis darbas, dideli kolektyvai, profesinė specializacija	Individualas darbas, maži kolektyvai
Griežtas planavimas, projekto valdymas	Neformalus planavimas, nevyksta projekto valdymas
Kokybės matavimai ir valdymas	Kokybė formaliai nėra vertinama
Žinios ir mokėjimai	Įgūdžiai ir talentas

Kodėl PS kūrimas tampa svarbiausia pramonės šaka? (5)

Nors PĮ yra nematoma, neapčiuopiama ir dažnai paslėpta, ji valdo visas moderniąsias technologijas ir daugybę įrengimų bei prietaisų – nuo pasaulinių telekomunikacijos tinklų iki skaitmeninių klausos aparatų. Programinių produktų gamybos apimtys auga eksponentiškai (viena iš priežasčių – vis greitesnė, mažesnių gabaritų ir pigesnė TĮ). PĮ slypi visur, valdo saugumo požiūriu kritines sistemas: *raketas, lėktuvus, automobilius, branduolinius reaktorius*. Tiek valstybės, tiek ir atskiros firmos mato, jog senąją ekonomiką keičia nauja, kurios pagrindas yra kompiuterinės technologijos.

Kokie svarbiausi pramoninio gamybos būdo ypatumai? (3)

- Pramoninis kuriamų gaminių pobūdis
- Darbo organizavimas
- Kruopštus projektavimas
- Aukštas darbo našumas
- Masinė gamyba
- Testavimas
- Standartizavimas
- Darbas pagal sandorius
- Procesų inžinerija
- Projekto valdymas
- Kokybės valdymas
- Komponentų naudojimas
- Santykis tarp pagrindinio ir aptarnaujančio kodo

Kuo pasireiškia pramoninis programų sistemų pobūdis? Kokie specifiniai tokių sistemų ypatumai? (2)

Taigi, pramoninės PS nuo kitų skiriasi šiais pagrindiniais aspektais

- galimybė keisti sistemos mastą *(scalability)*
- projekto trukmės, pačios sistemos ir vykdytojų grupės dydis
- struktūrinio sudėtingumo problemos svarbesnės už algoritminio sudėtingumo problemas
- būtina naudoti papildomas komunikavimo, projekto vadybos ir kokybės valdymo priemones
- padidinti sistemos saugumo reikalavimai
- nuolatinis tobulinimas
- PS šeimų kūrimas
- santykinai maža pagrindinio kodo dalis
- didelės apdorojamų duomenų apimtys
- profesionalūs interfeisai
- kolektyvinis sistemos naudojimas
- komunikavimas su kitomis sistemomis
- nuolatinis pasenusių sistemų keitimas naujomis

Ką reiškia pakeisti PS mastą? (1)

Didinti PS vartotojų skaičių, jos apdorojamų duomenų apimtis arba kokius nors kitus jos pajėgumus. Pramoninis kuriamų PS pobūdis pasireiškia tuo, kad jos platinamos pasauliniu mastu.

Kokios yra pagrindinės pramoniniu būdu kuriamų PS sudėtingumo priežastys? (1)

Pramoninėms PS būdingas kito pobūdžio sudėtingumas negu nepramoninėms(ne algoritmo sudėtingumas problema.. o jo matmenys.) Didelės programuotojų grupės, prireikia papildomų komunikavimo, projekto vadybos ir kokybės valdymo priemonių. Gaminio tvarkymas taip pat tampa sudėtingesniu, tenka naudoti profesionalius dokumentavimo bei konfigūracijos valdymo metodus.

Kaip suprantama pramoninių PS sauga? Kokiomis priemonėmis ji užtikrinama?(1)

Pramoninės PS, ypač kritinės sistemos, turi būti saugios. *(zero-defect)*

Kas yra permanentinė PS inžinerija? Dėl kokių priežasčių ji reikalinga? (1)

Daugelis gaminių nuolat tobulinami*(continuous engineering) – telefonų tinklai, buitinė elektronika, mikroelektronikos prietaisai, informacinės sistemos ir t.t.*.
Gana dažnai pramoninės PS ateina ne į tuščią vietą, bet keičia ankstesnes, liktines*(legacy)* sistemas. Veikiančių sistemų keitimas naujomis yra sudėtingas inžinerinis uždavinys apimantis:

- senos sistemos funkcijų perdavimą naujajai (migration),
- laipsnišką senos sistemos demontavimą (step-by-step rollout)
- kurį laiką besitęsiantį abiejų sistemų eksploatavimą (parallel operation).

Kas vadinama PS šeima? Kam tokių šeimų reikia?(1)

Paprastai yra kuriamos pramoninių gaminių šeimos*(product line)*. kuriami atitinkami tokių šeimų ruošiniai *(shared set of software assets)*. PS šeima vadinama grupė programų sistemų turinčių tarpusavyje daug bendrybių*(commonalities)*, tačiau besiskiriančių viena nuo kitos tam tikrais iš anksto žinomais skirtumais *(variabilities)*

Ką apima darbo organizavimo sąvoka? (1)

Darbo organizavimas apima atliekamų darbų struktūrizavimą, valdymą ir vykdymą. Jis taip pat apima institucinius darbo proceso aspektus: pareigybinę struktūrą, kas yra kieno viršininkas, kas kokias galias ir kokią atsakomybę turi, kaip reikia daryti konkrečius darbus, koks yra organizacijoje vykdomų užduočių pobūdis, įskaitant tų užduočių turinį ir apimtis.

Išvardinkite svarbiausias grupinio darbo problemas. (4)

- Žmonėms reikia bendrauti
- Jie turi dirbti kartu
- Juos reikia koordinuoti
- Laikas nuo laiko kažkas palieka grupę arba į ją ateina naujas asmuo

Kokie darbų pasidalijimo privalumai? (2)

Grupinis darbas tampa efektyvesniu, jo vykdytojai labiau specializuojasi konkrečioje srityje, pagal užduočių pobūdį, todėl geba dirbti našiau ir kokybiškiau.

Kokie darbų pasidalijimo ir vykdytojų specializavimo privalumai PS gamyboje? (2)

- vystomi specializuoti įgūdžiai
- taupomas laikas
- galima specializuoti įrankius.
- galima sukurti dideles ir sudėtingas sistemas.

Išvardinkite ir trumpai apibūdinkite Taylor'o sistemos elementus? (3)

Taylor'o sistemą sudaro 4 pagrindiniai elementai:

- Procesas suskaidomas į paprastas*(elementarias)* operacijas, atliekama sisteminė skaidymo būdo analizė, siekiant eliminuoti nebūtinas operacijas;
- Stebint geresnių nei vidutinių darbuotojų darbą, prižiūrint patyrusiems ekspertams, kiekvienai operacijai nustatomas jos vykdymo laikas;
- Jomis remiantis, nustatomos išdirbio normos. *(darbuotojai gauna premijas/baudas pagal jas)*
- Darbui prižiūrėti paskiriami funkciniai vadovai, besispecializuojantys pagal atskirus proceso aspektus ir atsakingi už tam tikrų operacijų grandinėlių vykdymą

Kokiais principais grindžiama šiuolaikinė darbo organizavimo teorija? (3)

- Darbuotojai turi būti motyvuojami ne bausmėm/paskatom, o moderniau *(pvz.: ugdant lojalumą firmai)*
- Darbuotojai turi būti traktuojami kaip asmenybės, turinčios poreikius, siekius ir vertybes, jų motivacija turi būti skatinama apeliuojant į jų savigarbą
- Reikia kurti glaudžiai sutelktas ir labai efektyvias darbuotojų grupes, suvokiančias save kaip organizacijos dalis ir pasiryžusias bet kuria kaina siekti organizacijos tikslų
- Grupių viduje turi viešpatauti bendradarbiavimo dvasia, grindžiama tarpusavio pagarba

Darbų pasidalijimas ir racionalus darbo organizavimas yra pagrindiniai skiriamieji pramoninės ir amatinės gamybos bruožai.

Kokie yra atsakomybės lygmenys grupiniame darbe? (1)

PS kurianti grupė apima visus asmenis, vienaip ar kitaip darančius įtaką projekto sėkmei arba nesėkmei. Atsakomybės lygmenys:

- ***bazinis:*** žmonės, tiesiogiai susiję su programinio produkto kūrimu;
- ***pagalbinis:*** žmonės, vienaip ar kitaip sudarantys darbo sąlygas bazinio lygmens darbuotojams;
- ***periferinis:*** žmonės, dirbantys ant projekto ribos *(pvz.: užsiimantys PS marketingu)*.

Kodėl, kuriant PS pramoniniu būdu, toks svarbus vaidmuo tenka dokumentavimui? (2)

- Į pramoninių PS programavimą įtraukta daug žmonių;
- Daug kitų žmonių, kurie patys nerašo kodo, taip pat dalyvauja tame procese, jas projektuodami, testuodami ar prižiūrėdami;
- Taigi, gera programos dokumentacija ir jos lengvai suprantama struktūra sutaupo daug laiko ir pinigų.

Kokias savybes privalo turėti geras, grupinį darbą dirbantis, darbuotojas? (2)

- Nuoširdžiai trokštantis projekto sėkmės;
- Suvokiantis, kaip jo elgsena atsiliepia į projekto sėkmę;
- Suprantantis savo vaidmenį: ką ir kodėl jam privalu daryti;
- Žinantis, ką ir kodėl dirba kiti;
- Gebantis objektyviai vertinti savo asmeninius privalumus ir trūkumus bei savo profesionalumo lygį;
- Linkęs, reikalui esant, padėti kitiems ir su jais bendradarbiauti;
- Mandagus ir pagarbus, tiek su užsakovais, tiek ir su bendradarbiais;
- Garbingas ir nekonfliktuojantis su bendradarbiais;
- Siekiantis turtinti kolektyvinę grupės patirtį.

Kokia darbo organizavimo patirtis pasiteisino ir gali būti rekomenduojama kitiems? (3)

- Viską nulemia žmonės monės – svarbiausias projekto resursas. Nevaržyk jų iniciatyvos.
- Aiškiai nustatyk, kas už ką atsako ir leisk savarankiškai priiminėti sprendimus jų atsakomybės ribose;
- Išsamiai išaiškink kolektyvui projekto tikslus ir ribojimus;
- Įsitikink, ar darbuotojai perprato naudojamą technologinį procesą ir standartus, kuriais turi vadovautis;
- Išaiškink darbuotojams, kur slypi pavojai nukrypti nuo nustatytų standartų.

Koks vaidmuo, kuriant PS pramoniniu būdu, tenka projektavimui? Kodėl projektavimas yra toks svarbus?(3)

Nelabai įmanoma sukurti pakankamai sudėtingą bet kokio pobūdžio sistemą, prieš tai jos nesuprojektavus. Išsiaiškinama ką gi norima sukurti. Įsitikinama, kad šitaip suprojektuota sistema iš tiesų veiks teisingai. Iš reikalavimų, kurie paprastai formuluojami sprendžiamos problemos terminais, paaiškėja tos problemos sprendimo būdai. Projektinė specifikacija aprašo į kokius komponentus turi būti suskaidyta sistema, kokie jų interfeisai. Padeda planuoti darbą.

Kas vadinama darbo našumu? (2)

Yra dvi darbo našumo sampratos:

- vieni vertina, kiek daug jūs padarėte;
- kiti skaičiuoja, kiek laiko jūs sugaišote ir kiek pinigų jums prireikė, kad padarytumėte tai, ką padarėte.

Kokius darbo našumo matavimo būdus jūs žinote? (3)

- ***Geriausias būdas:*** matuoti kolektyvo fondo grąžą. (darbas/laiko)
- ***Pramonėje:*** dažnai matuojama sukurtų produktų ir tam sunaudotų darbo vnt.*(žmonių, brigadų)* santykiu.
- ***Išeigos ir ieigos santykis:*** reikšmė, gauta palyginus tai, ką gavote, su tuo, ką įdėjote.
- ***Asmens parašytų kodo eilučių skaičius***.
- ***Laikas***, per kurį užsakovo poreikiai pertvarkomi į veikiantį ir išbandytą programų kodą.
- ***Gauto pelno***, *iš užsakovui pateiktos sistemos*, ***dydis vienam darbuotojui***.
- ***Užsakovo gauto pelno*** prieaugio ir vykdytojams sumokėtos sumos, ***santykis***.

Kokį poveikį darbo našumui turi darbo sąlygos? (4)

- Žmonės dirba našiau, jei darbo aplinka, darbo metodai ir jų naudojamos priemonės yra suprojektuoti jiems padėti.
- Jei jie dar bus ir skatinami, taip pat ir materialiai, gerai dirbti, greičiausiai jų darbo našumas išaugs.

Kaip padidinti PS kuriančių kolektyvų darbo našumą? (4)

- pasirinkti gerus žmones,
- neimti tinginių; netvarkingų; tų, kurie blogai mokėsi; tų, kurie kaitalioja darbo vietas; tų, kurie įsivaizduoja viską geriau už kitus išmanantys ir t.t.
- didinti kiekvieno technologinio proceso žingsnio efektyvumą,
- mažinti technologinio proceso žingsnių skaičių,
- eliminuoti perdirbinėjimus,
- mažinti kuriamos sistemos sudėtingumą,
- naudoti gatavus komponentus,
- Kadangi darbo našumas iš esmės priklauso nuo darbo aplinkos, tai galima pusiau rimtai teigti, kad jį galima padidinti tik tiek, kiek leidžia Microsoft. Alternatyva: Use open source ;)

Koks vaidmuo didinant PS patikimumą tenka instrumentinėms priemonėms? (1)

- Instrumentinėms priemonėms tenka lemiamas vaidmuo PS kūrimo projektuose.
    - Pagrindinis jų privalumas yra užduočių darbo imlumo sumažinimas.
    - Kita vertus, instrumentines priemones reikia perprasti ir įsisavinti.
- Darbai, kurie rankomis daromi savaitėmis, naudojant instrumentines priemones sutrumpėja iki valandų ar netgi minučių.
    - Augant darbų automatizavimo laipsniui, auga darbo našumas.
- Instrumentinių priemonių naudojimas padidina ne tik darbo našumą, bet ir kuriamo programinio produkto patikimumą (daugiau automatinio darbo).

Kokias programinių įrankių rūšis jūs žinote? (1)

- ***Programų konstravimo įrankiai:*** redagavimo sistemos*(editors)*, kompiliatoriai, generatoriai, derinimo stendai*(debuggers)* ir kt.
- ***PĮ kūrimo aplinkos****(SDE)*.

Kuo skiriasi programavimas mažuose ir dideliuose projektuose? (3)

Mažuose projektuose*(programming-in-the-small)*, visas programas rašo arba 1 asmuo ar nedidelė vienas kitą perpratusių asmenų grupė, sukuriamas kodas, kurį pajėgus perprasti vienas asmuo, bei naudojamos nesudėtingos programavimo aplinkos.

Dideliuose projektuose*(programming-in-the-large),* dirba nemažos programuotojų grupės, arba nedidelės programuotojų grupės dirba ilgą laiką ir sukurto kodo neįmanoma suvokti nepritaikius *"skaidyk ir valdyk"* principo. Todėl čia ypatingas dėmesys skiriamas sistemos skaidymui į modulius, darbą tenka kruopščiai planuoti bei dokumentuoti visą darbo procesą, o tam reikia specialių įrankių.

Kuo skiriasi *(programavimo)* įrankinės ir integruotosios *(programavimo)* aplinkos? (2)

Integruotos programavimo aplinkos*(IDE)* griežtai kontroliuoja įrankių naudojimą, visi įrankiai tarpusavyje integruoti, tuo tarpu įrankinėse vartotojas pats atsako už teisingą įrankių panaudojimą, o jungimas vieno su kitu daromas rankiniu būdu.

IDE pvz.: C++ aplinkos(Borland C++,CodeBlocks ir kt.) Ada aplinkos(IBM® Rational® Ada Developer ir kt.)
Įrankinių pvz.: Unix toolbox: Make, SCCS , SUN NSE (Network SE), Apollo DSEE (Domain Software Engineering Environment)

Kas vadinama CASE sistema(plačiau)? Pateikite pavyzdžių. (3)

- Paprastai CASE sistemos įrankiai palaiko visas arba bent jau daugumą technologinio proceso stadijų.
- Modernios CASE sistemos, be kita ko, palaiko grupinį darbą ir keletą sistemos kūrimo metodų.
- Vienas iš svarbiausių CASE sistemų privalumų yra tai, kad jos į PS kūrimo procesą įveda inžinerinės darbo disciplinos elementus.
- *Pvz.: Enterprise Architect 3.10, GDPro, Oracle Designer/2000, Rational Rose (IBM)*

Kas vadinama integruotomis projekto palaikymo aplinkomis(plačiau)? (3)

- IPPA numato priemones iš kitų šaltinių paimtiems įrankiams įjungti į jų sudėtį.
- Bene pagrindinis dėmesys IPPA skiriamas duomenų integravimo klausimams
    - integruoti duomenys panaudojami įrankiams koordinuoti;
    - IPPA repozitorijuje saugomi visi sistemoje naudojamų įrankių kuriami duomenys;
    - tradiciniai duomenų bazių valdymo mechanizmai repozitorijuose išplečiami mechanizmais, palaikančiais PS projektavimą bei konstravimą;
    - taigi, projekto duomenų bazė yra centrinis IPPA komponentas.

Kuo skiriasi CASE sistemos ir integruotosios projekto palaikymo aplinkos? (3)

CASE sistemos:

- ***Mechanizmai:***
  - Firminiai
  - Skirtingose sistemose bendrybių nedaug
  - Minimali semantika
- ***Paslaugos:***
  - Gausus asortimentas
  - Nestandartizuotos
  - Pritaikytos prie įrankių
- ***Procesas:***
  - Fiksuotas
  - Įrankių panaudojimo tvarka griežtai nustatyta

IPPA:

- ***Mechanizmai:***
  - IPPA karkaso interfeisai
  - Akcentuojamas duomenų integravimas
- ***Paslaugos:***
  - Skurdus asortimentas*(dažnai palaikomos tik horizontalios veikos, pvz.: konfigūracijos valdymas)*
- ***Procesas:***
  - Uždarose IPPA fiksuotas
  - Atvirose IPPA aprašomas

Kas bendro tarp masinės gamybos ir PS šeimų? (2)

Masinė gamyba ir masinis individualizavimas.

Kokį vaidmenį vaidina standartai pramoninėje gamyboje? (3)

Standardizavimas ir tiražavimas – pagrindiniai pramoninės ekonomikos šūkiai. Be standartų neįmanoma plačiai paskleisti technologijų, darančių didžiulę įtaką ištisiems pramonės sektoriams ir netgi nacionalinėms ekonomikoms.

Ką rekomenduojama dokumentuoti mažuose ir vidutinio dydžio projektuose? (3)

Rekomenduojami dokumentuoti mažuose ir kai kuriuose vidutinio dydžio projektuose dokumentai:

- ***Kontrakto modelis*** - aprašo sistemos arba jos komponento interfeisą. Skirtas subvykdytojams.
- ***Projektavimo sprendimai*** - tai yra svarbiausių projektavimo sprendimų, priimtų kuriant sistemą, santrauka ir tų sprendimų argumentavimas. Skirtas sistemą kuriančiam inžineriniam personalui, projekto vadovybei.
- ***Apžvalga vadovybei*** - sistemos vizija, ekonominiai skaičiavimai, laukiama nauda, darbo jėgos poreikis, darbų planas. Skirtas projekto/užsakovo vadovybei.
- ***Aptarnavimo dokumentacija*** - aprašo sistemos sąsajas su kitomis sistemomis, duomenų bazėmis bei kitais jos aplinkos elementais. Skirta aptarnavimo tarnyboms.
- ***Projekto apžvalga*** - projekto vizijos, jo kuriamų artefaktų aprašų, procesų aprašų ir pan. santrauka. Skirta inžinieriniam personalui, vadovybui, priežiūros tarnyboms, aptarnavimo tarnyboms.
- ***Reikalavimų specifikacija*** - aprašo ką sistema privalo daryti ir kokias kitas savybes ji turi turėti. Skirta inžinieriniam personalui, priežiūros tarnyboms, vartotojams, užsakovui.
- ***Diegimo dokumentai*** - diegimo tarnybų mokymo medžiaga (galimos problemos, inovaciniai slenksčiai, kontaktai ir kt.). Skirti sistemą diegiančiam personalui.
- ***Sistemos apžvalga*** - medžiaga, padedanti suprasti, kas tai per sistema, kokia jos paskirtis ir ką ji daro. Skirta priežiūros tarnyboms, projekto inžineriniam personalui.
- ***Vartotojo dokumentacija*** - sistemos žinynas, naudojimo instrukcija, galimos problemos ir jų sprendimo būdai, mokomoji medžiaga. Skirta vartotojams, jų vadovybei.

Kokie yra darbo pagal sandorius ypatumai? (3)

Pramoninių PS kūrimo darbus turi kas nors apmokėti. Didesnioji PS firmų dalis dirba pagal konkrečių užsakovų užsakymus, t.y. pagal konkrečius **sandorius** (kontraktus). Šitaip dirbanti firma yra vykdytojo vaidmenyje, o organizacija, užsakiusi kuriamąją PĮ ir finansuojanti jos kūrimo darbus, vadinama užsakovu. Vykdytojas, iš esmės, yra paslaugos tiekėjas ir todėl privalo įgyti užsakovo pasitikėjimą (ir tuo pačiu užsitikrinti sau darbą).

Dirbant pagal sandorį, sandoris pasirašomas tik po to, kai yra parengta išsami sistemos reikalavimų specifikacija (ji tampa sudėtine sandorio dalimi). Jei užsakovas panori keisti ar papildyti reikalavimus, yra rengiamas dokumentas "Reikalavimų specifikacijos pakeitimai" ir peržiūrimi projekto terminai bei pinigai. Perduodant užbaigtą PS sistemą užsakovui yra atliekami baigiamieji bandymai ir, jeigu nustatoma, kad sandorio reikalavimai neįvykdyti, ją tenka perdirbti (vykdytojo sąskaita). Nebaigus sistemos sandoriu nustatytu laiku, vykdytojui tenka mokėti baudas.

Sandoriai gali būti sudaromi, vadovaujantis įvairiais modeliais: fiksuoti terminai ir fiksuota kaina; išlaidos + fiksuotas pelno procentas; ir t.t. Tie modeliai gali būti įvairiai kombinuojami tarpusavyje.

Kas vadinama PS kūrimo technologiniu procesu? Ką apima proceso inžinerija? (2)

PS kūrimo technologinis procesas suprantamas kaip išsamios ir labai detalios taisyklės, nustatančios, kokiu būdu bus kuriamas, platinamas ir palaikomas programinis produktas, įskaitant visas jo dedamąsias dalis (modelius, dokumentaciją, užduotis ir planus, kodą, ruošinius ir t.t.). Technologinis procesas apima ir naudojamus programų sistemų kūrimo metodus. Metodai susistemina PĮ kūrimo ir priežiūros veiklas.

Taisyklės nustato darbų eiliškumą, naudojamus instrumentus, pasiskirstymą darbais, darbo metodus, vykdytojų kvalifikacinius reikalavimus ir visus kitus technologinių operacijų vykdymo aspektus.

Proceso inžinerija apima tokių taisyklių sudarymą, jų įgyvendinimą, proceso parametrų matavimus ir proceso tobulinimą tų matavimų pagrindu.

Į ką privalo atsižvelgti technologas, konstruodamas PS kūrimo technologinį procesą? (3)

Projektuodamas technologinį procesą, technologas privalo atsižvelgti į:

- projekto trukmę,
- dirbančiųjų skaičių,
- jų geografinį (taip pat ir pastato viduje) išsidėstymą,
- prognozuojamą reikalavimų keitimo dažnį,
- prieš tai naudoto technologinio proceso brandą,
- projekto biudžeto dydį,
- vykdytojų išsilavinimą, darbo kultūrą, įpročius bei tradicijas,
- vykdytojų motyvavimo sistemą.

Kokie yra PS kūrimo projektų tipai? (1)

PS inžinerijos projektų tipai :

- ***naujos sistemos kūrimas*** – tokiame projekte niekas neriboja*(proto ribose)* nei sistemos reikalavimų, nei projektavimo sprendimų.
- ***esamos sistemos reinžinerija*** – kai jau yra naudojama PS, tačiau ji dėl nuolatinių keitimų *"susidėvėjo"*.
- ***esamos sistemos priežiūra****(maintenance)* – sistemos tobulinimai ar keitimai.
- ***programų paketo įsigyjimas****(package selection)*.
- ***esamos sistemos kėlimas į kitą platformą****(system conversion)*.

Ką apima projekto valdymas? (1)

Projekto valdymas apima produkto valdymą, proceso valdymą, resursų valdymą ir aplinkos valdymą.

Ką apima produkto valdymas? (4)

Produkto valdymas apima versijų tvarkymą, konfigūracijos kontrolę, produkto atmainų (release) valdymą, produkto komponentų trasavimą.

Versijų kontrolė:

- versijų komponentų priežiūra,
- versijų paieška ir parinkimas,
- iš versijos į versiją keliamų komponentų pakeitimų koordinavimas

Konfigūracijos kontrolė:

- sistemos modeliavimas,
- sistemos konstravimas,
- išvestinių objektų tvarkymas - objektų, gautų iš kitų objektų panaudojus atitinkamus įrankius (pvz., kodas sugeneruotas iš UML diagramų) priežiūra, efektyvi kainos požiūriu.

Produkto atmainų valdymas:

- bazinių komplektų tvarkymas (baselining) ir produkto dalių pakavimas - pvz, vykdomasis kodas pakuojamas kartu su vartotojo dokumentacija.
- produkto atmainų priežiūra ir informacijos apie jas teikimas klientams,
- pasenusių atmainų demontavimas.

Produkto komponentų trasavimas turi užtikrinti:

- sąryšius tarp dokumentų, aprašančius produkto komponentus,
- tų sąryšių trasavimą per versijas, atmainas ir t.t.,
- dokumentų paiešką pagal tuos sąryšius.

Ką apima procesų valdymas? (4)

Procesų valdymas užsiiminėja disciplinuoto ir "tvarkingo" PS kūrimo proceso standartais, procedūromis ir protokolais. Jis apima pokyčių valdymą, kokybės kontrolę ir užduočių vadybą.

Pokyčių valdymas užsiima realizuotų sistemos versijų keitimais ir su tuo susijusiomis ataskaitomis apie klaidas.

Kokybės kontrolė standartų ir vertinimo bei verifikavimo procedūrų forma nustato kriterijus, kurie turi būti tenkinami atitinkamais proceso laiko momentais.

Užduočių vadyba apima užduočių priskirimą vykdytojams, darbų nustatymą, kuriuos, *vykdant užduotis*, privalo atlikti užsakovas, veiklų pabaigos fiksavimą ir ataskaitų apie tai rengimą, susijusių užduočių aktyvavimą.

Ką apima aplinkos valdymas? (2)

Aplinkos valdymas užsiima vadybinėmis problemomis, susijusiomis su projekto technologinės aplinkos pokyčiais, įskaitant TĮ pokyčius.

Ką apima resursų valdymas? (4)

Resursų valdymas apima su projekto planavimu ir vykdymo kontrole susijusius darbus, įskaitant

- planavimą - projekto skaidymą į užduotis, finansinį planavimą, resursų priskyrimą užduotims, kalendorinio darbų plano sudarymą;
- monitoringą - faktinių vykdymo terminų ir faktinių resursų sąnaudų fiksavimą ir lyginimą su planuotais terminais bei planuotomis resursų sąnaudomis
- perplanavimo darbus - atliekami pasikeitus darbuotojų sudėčiai, organizacijos struktūrai ar kuriamo programinio produkto struktūrai, o taip pat, nukrypus nuo planuotų terminų ar kitų planinių rodiklių.

Koks vaidmuo PS kūrimo projektuose tenka neformaliam bendravimui? (3)

Organizacijose, kuriose dažnai kinta technologinė aplinka arba vykdomi "nestabilūs" projektai, ypač svarbus vaidmuo tenka **neformaliam bendravimui**. Dažnai kintančiose technologinėse aplinkose tenka nuolat peržiūrėti ir keisti sistemos reikalavimus, o formalūs bendravimo mechanizmai (per dokumentus) tokiose situacijose yra per lėti. Panaudojant neformalius bendravimo mechanizmus, informacija apie pokyčius, jų svarbą ir galimas jų pasekmes paskleidžiama daug greičiau. Vienu iš efektyviausių neformalaus bendravimo mechanizmų yra Web serveriai.

Kokių tikslų siekiama kokybės užtikrinimo procesu? (1)

PĮ kokybės užtikrinimo procesas siekia užtikrinti, kad programiniai produktai atitiktų reikalavimus ir kad tų produktų kūrimo procesai vyktų pagal sudarytus planus.

Kokia veikla vadinama PĮ testavimu? Kaip testuojami atskiri PS komponentai? (2)

Testavimas tai netokia veikla,kuri pradedama tik baigus kodo rašymą ir kuria siekiama tik išsiaiškinti, ar sistema "nenusimušinėja". PĮ vyksta viso PS kūrimo proceso metu ir juo siekiama visapusiškai kontroliuoti visų kuriamų artefaktų kokybę. Testavimą pradedama planuoti jau formuluojant sistemos reikalavimus ir po to, projektui progresuojant, testavimo planai ir procedūros yra nuolat tobulinami.

Kuo svarbus daugkartinis programų kodo panaudojimas? (2)

Pakartotinis komponentų panaudojimas eliminuoja bereikalingą "dviračio išradinėjimą". Jis taupo laiką ir leidžia programuoti ne "nuo nulio", naudojantis jau anksčiau kitų sukurtais komponentais. Pramoninė PS gamyba daugkartinį kodo panaudojimą pavertė visuotine praktika.

Programinių komponentu panaudojimas ir pavyzdžiai. (3)

- interfeisų apibrėžtis ir jų administravimo taisykles nustato konkreti komponentų saugykla (repozitorijus),
- prieidamas prie komponento interfeisų, klientas gauna informaciją apie to komponento teikiamas paslaugas; to komponento elgsenos pasekmes klientas patiria jį aktyvuodamas,
- toks būdas leidžia komponuoti sistemas dinamiškai, pagreitina PS kūrimą, jas atpigina, padidina jų patikimumą ir lankstumą.

Pvz: komponentas "Užsakovas" turi 2 paslaugas - "Informacija apie užsakovą", "UžsakovoSąskaita" ir 2 interfeisus joms pasiekti. Vieno interfeiso įeiga: UžsakovoKodas, išeiga - informacija apie užsakovą. Kito interfeiso įeiga: UžsakovoKodas, išeiga: užsakovo sąskaitos detalės.

Komponentinė PS architektūra. (3)

- Klientas <-> Objektų reikalavimų brokeris
- Objektų reikalavimų brokeris <-> 1-as komponentų serveris *(1 paslauga, 2 paslauga)*
- Objektų reikalavimų brokeris <-> 2-as komponentų serveris *(1 paslauga, 2 paslauga)*

T.y.: klientas bendrauja su objektų reikalavimų brokeriu – jis randa klientui reikiamas paslaugas pagal turimą komponentų serverių sąrašą, bei grąžina informaciją kaip bendrauti su tų paslaugų komponentų interfeisais.

Pavyzdžiai kalbų, tinkančių interfeisų aprašymams? (2)

CORBA komponentams interfeisų aprašymo kalba yra C++. Dar tam tinka XML kalba.

Kokias komponentines technologijas jūs žinote? (2)

- Microsoft' COM / DCOM / COM+ / .NET/ C#
- Sun' JavaBeans, Enterprise Java Beans (EJB)
- OMG' Component Model (CCM)
- OSGI

Kuo skiriasi pagrindinis ir visas PS kodas? (1)

Pagrindinis PS kodas - kodas, vykdantis funkciniuose sistemos reikalavimuose numatytas funkcijas. Visas PS kodas savyje turi ir pagalbinį kodą, skirtą duomenų analizei, klaidų paieškai, žurnalizacijai, vartotojų autorizavimui, informavimo sistemoms *(help)* ir kitiems nefunkciniams reikalavimams įgyvendinti.

Kaip skiriasi pagrindinio ir viso kodo santykis pramoninėse ir nepramoninėse PS? Kokią įtaką tai turi pramoniniam PS kūrimo procesui? (3)

Nepramoninėse PS didžioji kodo dalis yra pagrindinis kodas.

Pramoninėse PS pagrindinio kodo dalis yra apie 25-50%. Kitą kodo dalį sudaro pagalbinis kodas. Tai reiškia, kad tik nuo pusės iki trijų ketvirtadalių viso kodo rašoma ne pačiam uždaviniui spręsti, o užtikrinti kitoms pramoniniam gaminiui būtinoms savybėms. Tokio santykio pasekmės - tampa tikslinga universalizuoti įvairias pagalbines funkcijas (klaidų apdorojimas, trasavimas, vartotojų atpažinimas ir kt.) ir kurti jas taip, kad jas būtų galima panaudoti daugelyje sistemų, vienu ar kitu būdu, geriausiai automatiškai, įterpiant pagalbinio kodo gabalus į atitinkamas pagrindinio kodo vietas.

Kaip skiriasi apdorojamų duomenų apimtys pramoninėse ir nepramoninėse PS? (1)

Pramoninės PS apdoroja dideles duomenų apimtis. Su pramoninėmis PS, ypač Web sistemomis, gali dirbti didelės vartotojų grupės. Dėl to tenka naudoti sudėtingas DBVS, transakcijų monitorius bei įvairią tarpinę PS.

Nepramoninės PS apdoroja daug mažesnius duomenų kiekius.

5 paskaita

21. Į kokias grupes yra skirstomi PS inžinerijos procesai? Trumpai apibūdinkite kiekvieną iš jų. (2)

PS inžinerijoje naudojami 3 tipų procesai:

- ***baziniai****(pirminiai)* ***procesai*** – be jų neįmanoma nei sukurti jokią PS sistemą, nei ja pasinaudoti
- ***pagalbiniai procesai*** – veikia bazinių sudėtyje, padeda užtikrinti jų kokybę bei sėkmę, bet nėra būtini
- ***organizaciniai procesai*** – jais kuriamos, aptarnaujamos ir palaikomos organizacinės struktūros, skirtos PS kurti

22. Kokie PS inžinerijos procesai yra vadinami baziniais procesais? Trumpai apibūdinkite kiekvieną iš jų. (3)

Baziniai procesai:

- ***analizė:*** ko nors esamo*(proceso, veiklos, sistemos ir kt.)* modeliavimas, atliekamas tikslu tai perprasti;
- ***specifikavimas:*** ko nors, kas bus kuriamas, modeliavimas jo iš išorės stebimų savybių terminais, atliekamas tikslu suformuluoti to daikto projektavimo ir realizavimo tikslus ir ribojimus;
- ***projektavimas:*** ko nors, kas bus kuriamas, modeliavimas jo vidinių savybių terminais, atliekamas tikslu kuo geriau įgyvendinti specifikacijos reikalavimus, projektavimas apima geriausio specifikacijos įgyvendinimo būdo parinkimą ir to būdo modeliavimą;
- ***realizavimas:*** ko nors, kas yra kuriama, specifikuotą to dalyko elgseną projektuotojo numatytu būdu įgyvendinančių dalių konstravimas:
    - ***programavimas*** – vienas iš subprocesų, suprantamas kaip projektavimo sprendimų užrašymas pasirinktam procesoriui aiškia instrukcijų seka;
- ***surinkimas****(integravimas)*: ko nors, kas yra kuriama, dalių jungimas į vieną visumą;
- ***diegimas****(preparation for operation)*: kam nors veikti reikalingų resursų akumuliavimas, to daikto pateiktis, instaliavimas ir pritaikymas konkrečiai darbo vietai;
- ***aptarnavimas****(operation)*: specifikacija numatytos kieno nors elgsenos palaikymo procesas;
- priežiūra*(maintenance)*: ko nors, kas jau yra sukurta ir naudojama, vidinių ar iš išorės stebimų savybių keitimo procesas, atliekamas nepažeidžiant to daikto vientisumo ir jo savybių darnos;
- ***demontavimas****(retirement)*: ko nors sukurto ir naudojamo laipsniškas naudojimo nutraukimas

PS atveju – tai laipsniškas vienos veikiančios sistemos funkcijų perdavimo kitai veikiančiai sistemai procesas.

23. Kokie PS inžinerijos procesai yra vadinami pagalbiniais procesais? Trumpai apibūdinkite kiekvieną iš jų. (3)

Pagalbiniai procesai

- ***testavimas****:* empirinis tikrinimas, ar kas nors tenkina reikalavimus, kuriuos tas daiktas privalo tenkinti,
- ***verifikavimas***: ko nors, kas yra kuriamas, eilinio atlikto kūrimo žingsnio (pvz., operacijos) korektiškumo tikrinimo procesas,
- ***vertinimas*** (validation): procesas, kuriuo siekiama nustatyti, ar kas nors tenkina realius to daikto vartotojų poreikius,
- ***dokumentavimas***: informacijos, sukauptos ką nors kuriant, fiksavimo procesas,
- ***konfigūracijos valdymas****:* procesas, kuriuo siekiama, kad kas nors, kas yra kuriamas ir nuolat keičiamas, bet kuriuo laiku momentu turėtų visas savo sudėtines dalis, privalomas tam laiko momentui, ir kad tos dalys tarpusavyje būtų suderintos,
- ***peržiūra*** (reviewing): procesas, kuriuo siekiama patikrinti, ar ko nors, kas yra kuriamas, kūrimas vyksta pagal numatytą planą, ar gauti visi plane numatyti rezultatai ir, jei to nėra, nustatyti nukrypimų nuo planuotos darbų eigos priežastis,
- ***inspektavimas*** (auditing): procesas, kuriuo siekiama patikrinti, ar ką nors kuriant nenukrypstama nuo sandoriu, standartais ar kitais privalomais dokumentais nustatytų kokybės reikalavimų,
- ***problemų sprendimas*** (problem resolution): procesas, kuriuo siekiama išanalizuoti problemas (įskaitant nukrypimus nuo planuotos darbų eigos), nustatytas atliekant peržiūrą, inspektavimą ar kitus procesus, tų problemų prigimtį bei priežastis, ir tas problemas pašalinti.

24. Kokie PS inžinerijos procesai yra vadinami organizaciniais procesais? Trumpai apibūdinkite kiekvieną iš jų. (3)

Organizaciniai procesai:

- ***valdymas***: kitų procesų iniciavimas, planavimas, vykdymas, kontrolė, vertinimas ir užbaigimas,
- ***infrastruktūros kūrimas ir palaikymas***: kitiems procesams vykdyti reikalingų sąlygų sudarymo procesas,
- ***procesų tobulinimas***: kitų procesų matavimo, kontrolės vertinimo ir gerinimo procesas,
- ***apmokymas*** *(training)*: procesas, kuriuo siekiama, kad kitus procesus vykdantys asmenys įgytų tam reikalingas žinias bei įgūdžius.

25. Kokios artifaktų grupės yra kuriamos PS inžinerijos procesais? Trumpai apibūdinkite kiekvieną iš jų. (3)

PS inžinerijos procesais kuriamos šešios artefaktų grupės:

- ***preliminariniai dokumentai***: tikslų specifikacijos, poreikių specifikacijos, verslo modeliai ir t.t.;
- ***techniniai sistemos aprašai***: reikalavimų specifikacijos, projektinės specifikacijos, kodas, vartotojo dokumentacija;
- ***sistemos mazgai***: posistemiai, moduliai, duomenų bazės ir t.t.;
- ***sistemos ruošiniai****(distributive versions)*: tai, iš ko generuojamos veikiančios sistemos;
- ***galutiniai programiniai produktai***: veikiančios sistemos;
- ***pagalbinė medžiaga***: planai, užduočių aprašai, ataskaitos ir t.t.

26. Kokios idealizuotos prielaidos daromos PS inžinerijoje? (3)

PS inžinerijoje, kaip ir kitose mokslinėse disciplinose, daromos tam tikros idealizuotos (neatitinkančios tikrovės) prielaidos, apie tai, kaip PS inžinerijos procesai vyksta praktikoje:

- galima "*įšaldyti*" sistemos reikalavimus,
- galima sukurti neturinčius klaidų artefaktus,
- galima sumodeliuoti sistemos elgseną trykių metu,
- įmanoma surinkti pakankamai duomenų artefaktams vertinti.

33. Kas yra procedūrinė abstrakcija? (3)

Procedūrinė abstrakcija – tai situacija, kai moduliai suvokiami kaip procedūros.
*Pvz. kėlimas kvadratu – vietoje to, kad kiekvieną kartą rašyti išraišką x*x, paimkime paimkime funkcinę juodąją dėžę(ji realizuojama procedūra), kurios įeiga yra skaičius, o išeiga – to skaičiaus kvadratas.*

- Funkcija – viena iš svarbiausių skaičiavimo mokslinių idėjų
  - Funkcija inkapsuliuoja kokį nors algoritmą.
  - Tuo algoritmu programoje galima pasinaudoti iškviečiant funkciją (darant nuorodą į jos vardą) ir pateikiant reikalingus pradinius duomenis (funkcinės dėžės įeigą).
- *Procedūrinės abstrakcijos privalumai*:maskuojamos algoritmo detalės, palengvėja kodo skaitomumas, sudaroma galimybė galvoti apie bendrą algoritmo schemą, atidedant jo realizavimo detales vėlesniam laikui, sukuriami daugkartinio panaudojimo komponentai*(funkcijų bibliotekos)*, kuriais galima pasinaudoti ir kitose programose, sudaroma galimybė tobulinti funkcijos realizaciją. Nedarant jokių kitų pakeitimų programoje, vardai lokalizuojami funkcijos viduje, projektuotojui suteikiama galimybė projektuoti funkcinių (leksinių) agregatų terminais, palengvėja PS šeimų projektavimas.

Taisyklė:

- Apibrėžk formalius parametrus, parašyk atitinkamą algoritmą realizuojančios procedūros kūną.
- Suteik procedūrai vardą.
- Paslėpk algoritmines detales nuo vartotojo, leisdamas jam kviesti procedūrą pagal vardą.

34. Kas yra duomenų abstrakcija? (3)

Duomenų abstrakcija – tai sudėtingos duomenų struktūros grindžiamos turinių atskyrimo principu: griežtai vienos nuo kitų yra atskiriamos abstrakčiosios DT savybės ir konkrečios to tipo realizavimo detalės.

- Abstrakčiasias savybes turi žinoti tas, kas tuo duomenų tipu nori pasinaudoti (duomenų tipo interfeisas). Realizavimo jam žinoti nereikia, jis gali kisti, pvz. padidinus programos našumą.
- ADT keitimai yra lokalūs, nedaro poveikio likusiai programos daliai, nes nekeičia ADT elgsenos.
- Duomenų abstrakcija duoda galimybe agreguoti duomenis ir dirbti tų agregatų terminais
    - duomenų agregatas gali būti vėl traktuojamas kaip primityvus objektas ir agreguojamas kartu su kitais objektais, šitaip kuriant aukštesnio lygmens duomenų agregatus.

Procedūrinėmis ir duomenų abstrakcijomis grindžiamas objektinis projektavimas. Čia leidžiama į vieną visumą agreguoti funkcinius ir duomenų agregatus. Taip kuriamos objektinės juodosios dėžės*(objektai)*, realizuojančios vartotojo apibrėžtus probleminius DT.

- OP teorijoje, abstrakcija suprantama kaip galimybė apibrėžti objektus, representuojančius abstrakčius "aktorius", galinčius atlikti darbą, pranešti/keisti savo būseną, „*komunikuoti*" su kitais tos PS objektais
- Yra skirtumas tarp ADT ir jam realizuoti naudojamos duomenų struktūros.
    - Pvz: ADT sąrašą galima realizuoti masyvu arba susiejant sąrašo elementus nuorodomis.
    - ATD sąrašas turi tiksliai apibrėžtas *operacijas (pridėti/išmesti elementą ir t.t.)*, o nuorodomis susieti sąrašo elementai*(sąrašinė duomenų struktūra)* yra viena iš duomenų struktūrų, kurią panaudojant, galima gana paprastai tas operacijas realizuoti.
    - Kadangi ATD sąrašas dažnai yra taip realizuojamas, tai pradedama tuos dalykus painioti ir pradedama sąrašines duomenų struktūras vadinti sąrašais. Tačiau tai yra skirtingi dalykai.
- Duomenų struktūros konstruojamos remiantis sandorio principais*(įsipareigojama nekeisti interfeiso)*. O su jomis yra susiejamos procedūros, skirtos jomis manipuliuoti
- Paprastai rekursyviosios procedūros slepia rekursyviasias duomenų struktūras

35. Kas yra valdymo abstrakcija? (3)

Valdymo abstrakcija – tai aukšto lygmens programavimo kalbų valdymo struktūros:

- Mašininėse kalbose valdymo struktūros yra labai primityvios ir konkrečios.
- FORTRAN, BASIC ir kitos ankstyvosios progr. kalbos, mėgdžiojo mašininių kalbų valdymo struktūras.
- Neturint abstraktesnių valdymo struktūrų, programose daroma daug klaidų ir negalima adekvačiai aprašyti programos operatorius siejančius loginius ryšius.

<u>Valdymo abstrakcijos pavyzdžiai:</u>

- ***Šakojimo struktūros***: *if-then*; *if-then-else*; *switch* ir t.t.
- ***Ciklai:*** *while*; *do-while*; *for* ir t.t.
- ***Valdymo perdavimas:*** *goto*; *exit*; *return*; *break*; *continue*; *call* ir t.t.

36. Kas yra, kaip kurima virtualioji mašina(plačiau)? (3)

- Kuriamos panaudojant realią aparatūrą. Iš karto gali veikti kelios tos pačios mašinos kopijos, nepriklausančios viena nuo kitos.
- Leidžia programoms ignoruoti TĮ ypatumus ir vykdyti tas programas skirtingose platformose, jei jos emuliuoja tą pačią virtualią mašiną. Taip viename kompiuteryje gali egzistuoti, tarpusavyje nesuderinamos, skirtingos programavimo/skaičiavimo aplinkos.
    - Virtualioji mašina realizuojama programiškai, šitaip paprastai realizuojamas apatinis PS abstrakcijos lygmuo.
        - Virtualioji mašina realizuojama panaudojant kitas abstrakcijas (procedūrinę, duomenų, valdymo, įrenginio). Ji nuslepia nuo PS konkrečios platformos detales.
        - Keliant PS į kitą platformą, pakanka perrašyti tik jos virtualiąją mašiną.
        - Aukštesni lygmenys nekinta, nes nekinta abstrakčiosios mašinos interfeisai (jos API).
- *VM Pvz.: Assemblerio mašina, C++ mašina, Buhalterinių skaičiavimų mašina*

Reziumė:

- VM yra programiškai sukuriama fikcija – nebūtinai realią, gali būti ir kita abstrakti mašina.
- Kol VM reaguoja į programos veiksmus kaip reali mašina*(išlieka juodąja dėže)*, ją galima traktuoti kaip realų kompiuterį.
- VM ir realaus kompiuterio interfeisą galima traktuoti kaip API.
    - Šį netradicinį API sudaro pertraukimai, kreipiniai į BIOS ir I/O prieigas (ports).

46. Palyginkite “iš viršaus žemyn” ir “iš apačios aukštyn” paradigmas. (2)

- Rekomenduojama vadovautis “iš apačios aukštyn” paradigma, kuomet norima sukurti PS sprendžiančią bendresnę problemą negu problema, kurią sprendžia kokia nors jau esama PS.
- Kaip jau minėta, svarbiausias paradigmos trūkumas yra tas, kad apatiniuose abstrakcijos lygmenyse priimti technologiniai sprendimai paveikia visą PS, įskaitant vartotojo interfeisus
    - vadovaujantis “iš viršaus žemyn” paradigma, pradedama ne nuo technologinių sprendimų, jie atidedami vėlesniam laikui.
- Lipant “iš viršaus žemyn” galima nenulipti į turimą kompiuterinę platformą. Lipant “iš apačios aukštyn” galima neužlipti į sprendžiamą problemą. Todėl praktikoje dažniausiai abi paradigmos yra derinamos viena su kita.

6 paskaita

1. Iš kokių dalių susideda *reikalavimų inžinerija*? (1)

Reikalavimų inžinerija apima:

- **poreikių analizę;**
- **reikalavimų analizę;**
- **reikalavimų specifikavimą.**

3. Paaiškinkite *operacinių reikalavimų* ir *sistemos reikalavimų* skirtumus. (1)

- ***Operaciniai reikalavimai*** išplaukia iš vartotojų veiklos poreikių*(operacinių poreikių)*:
  - Bendraudamas, analitikas privalo išsiaiškinti ir dokumentuoti, vartotojų ir užsakovų poreikius, lengvai suvokiama forma – vartotojams, užsakovams, vykdytojo vadovybei ir PS inžineriams.
- ***PS reikalavimai*** išplaukia iš operacinių reikalavimų:
  - Analitikas privalo „*transformuoti*“ poreikių specifikaciją į PS reikalavimų specifikaciją ir suformuluoti reikalavimus taip, kad jie būtų lengvai suvokiami tiek vartotojams ir užsakovams, tiek vykdytojo organizacijos vadovybei, tiek ir PS inžinieriams.

4. Paaiškinkite, kuo svarbūs *operaciniai reikalavimai*. (2)

Tai vartotojų reikalavimai sistemai, kuriuos reikia perduoti tiems, kas kur tą sistemą.

5. Kokiu tikslu rengiamas dokumentas „*Verslo poreikių specifikacija*“ ir kas jame rašoma? (2)

- VPS paskirtis yra aprašyti pačias bendriausias operacines būsimos sistemos savybes – t.y. kaip ja bus naudojamasi versle ir ką verslas iš to laimės.
- Dokumentas naudojamas visų projekto dalyvių *(vartotojų, užsakovo, vykdytojų ir kt.)* siekiamiems tikslams koordinuoti bei derinti.

7. Kas tai yra *SWOT analizė*? Kokiu tikslu ji atliekama? (2)

Išorinės ir vidinės aplinkos analizė yra svarbi vizijos formulavimo ir strategijos planavimo proceso dalis:

- verslo požiūriu, ***vidiniai aplinkos veiksniai***, dažnai skirstomi į verslo stipriąsias ir silpnąsias puses;
- verslo požiūriu, ***išoriniai aplinkos veiksniai***, dažnai skirstomi į galimybes ir grėsmes.

SWOT siekiama išsiaiškinti teigiamas/neigiamas išorinės aplinkos tendencijas, galinčias paveikti verslo proceso efektyvumą.

8. Kas vadinama *stipriosiomis verslo pusėmis*? (1)

Tai jo turimi resursai ir galimybės, kuriomis galima pasinaudoti siekiant konkurencinių pranašumų. ***Pvz.:***

- patentai;
- įtvirtinti brendai;
- gera reputacija tarp klientų;
- firminės (kitiems nežinomos) technologijos, leidžiančios ką nors padaryti pigiau;
- geroje vietoje (pavyzdžiui, miesto centre) esanti buveinė.

9. Kas vadinama *silpnosiomis verslo pusėmis*?(1)

Stipriųjų pusių nebuvimas traktuojamas kaip silpnoji pusė, pavyzdžiui:

- veikla neapsaugota patentais;
- nėra įtvirtintų brendų (liet. *prekinių vardų, ženklų*);
- bloga reputacija tarp klientų;
- aukšta gamybos savikaina;
- blogoje vietoje esanti buveinė.
- nebuvimas galimybių naudotis svarbiausiais pardavimo tinklais

2. Paaiškinkite *išorinės analizės* esmę.(3)

Išorinės analizės metu visas dėmesys sutelkiamas į verslo proceso aplinką:

- kokios yra pagrindinių varžovų vizijos?
- ar analizuojamo verslo vizija sugretinama su jo varžovų vizijomis?
- jei ne: ar yra svarbūs argumentai pasirinkti visiškai skirtingą viziją?

3. Kokie *verslo aspektai* analizuojami atliekant išorinę analizę? Trumpai apibūdinkite kiekvieną iš aspektų. ~(5)

Keturi analizės matmenys:

- ***įeigos logistika*** – tiekėjai, tiekimo problemos, tiekimo grėsmės;
- ***išeigos logistika*** – platintojai, perpardavinėtojai, užsakovai, klientai, platinimo bei pardavimo problemos ir grėsmės;
- ***teisinis reguliavimas*** – teisiniai ir privatumo apsaugos klausimai, su jais siejamos problemos ir grėsmės;
- ***išorinis vertinimas****(įvaizdis)* – vertintojai, įvaizdžio problemos ir grėsmės;

4. Paaiškinkite *vidinės analizės* esmę.(3)

Vidinės analizės metu sprendžiama, kokie procesai ir kokiais aspektais yra susiję su siekio įgyvendinimu, bei nustatoma, kas ir kokiu mastu turi būti analizuojama. Stengiamasi atsakyti į šiuos klausimus:

- kokia yra funkcinė analizuojamos dalykinės srities struktūra ir kaip tos funkcijos tiesiogiai ar netiesiogiai palaiko suformuluotą strategiją;
- kokie vidiniai procesai naudojami funkcijoms realizuoti;
- kokios užduotys atliekamos vykdant vidinius procesus;
- kokie agentai organizuoja analizuojamų procesų vykdymą, kokia atsakomybė tenka kiekvienam iš jų ir kokias užduotis jie atlieka;
- kaip keisis tų agentų veiksmai (kokią naudą jie turės) įdiegus kuriamąją programų sistemą.

5. Paaiškinkite, kas tai yra *penkios Porter'io jėgos*. (3)

Penkios Porter'io jėgos:

- modelis nagrinėja verslo 'pajėgumą' ir artimiausias atakas, galinčias sugriauti tą verslą;
- Trumpas 'įtariamųjų' sąrašas yra toks:
    - varžovai;
    - klientai ir užsakovai;
    - tiekėjai;
    - 'pakaitalus' tiekiančios bendrovės;
    - bendrovės 'galinčios išstumti' verslą ir kaip galima nuo jų apsisaugoti*(kaip stabdyti jų plėtrą)*.

6. Koks vaidmuo išorinėje analizėje tenka *sėkmės matams*? Pateikite tokių matų pavyzdžių. (4) (atsakyta, manau, nepilnai)

- Verslo sėkmės matai naudojami verslo aspektams (veiksniams) matuoti, norint nustatyti, kiek sėkmingai tas verslas vyksta
- Išorinės analizės metu verslo problemos aptinkamos atliekant to verslo efektyvumo matavimus ir vertinimus.
- Stengiamasi atsakyti į šiuos klausimus:
    - Kiek sėkmingai vyksta verslas?
    - Ar yra pasiekiami verslo tikslai?

7. Paaiškinkite, kaip traktuojamos *verslo stipriosios ir silpnosios pusės vidinės analizės* metu. (3)

Nagrinėjant stipriąsias puses, galima rasti nepanaudotą potencialą bei nustatyti tas turimas kompetencijas, kurios praeityje prisidėjo prie verslo sėkmės.

Nagrinėjant silpnąsias puses, galima rasti našumo spragas, pažeidžiamas (silpnas) vietas bei neteisingas esamų strategijų prielaidas (t.y., problemų priežastį).

8. Kokie *metodai* naudojami *vidinei analizei atlikti*? (2)

- Išteklių auditas;
- kaštų ir pelno analizė;
- lyginamoji analizė (benchmarking);
- vertės grandinės analizė;
- tiekimo grandinės analizė.

9. Koks strategijos ir vizijos ryšys? (2)

Strategijos priemonė, principų rinkiniu, siekiama išpildyti viziją.

10. Kokius *strategijų tipus* žinote? Trumpai juos apibūdinkite. (2)

- **Korporacijos lygmens strategijos:** nusako kaip siekiama pertvarkyti visą verslą – tai ilgalaikės ir labai bendros strategijos.
- **Konkuravimo strategijos**: nusako kokiu būdu bus varžomasi su konkurentais.
- **Funkcinio lygmens strategijos**: nusako ko turi siekti organizacijos funkciniai padaliniai *(buhalterija, atsiskaitymų skyrius ir t.t.)*

1. Kokias *konkuravimo strategijas* žinote? Trumpai jas apibūdinkite. (3)

Bendrosios Michael Porter'io strategijos:

- **kaštų lyderio strategija:** siekiama turėti kuo mažesnius gamybos bei pardavimo kaštus ir pardavinėti gaminius bei paslaugas pačiomis mažiausiomis kainomis;
- **išskirtinumo strategija:** siekiama teikti unikalius, vartotojų ypač vertinamus ypatumus turinčius gaminius bei paslauga;
- **fokusavimo strategija**: kaštų lyderio arba išskirtinumo strategija, taikoma kokiam nors siauram rinkos segmentui (specialaus pobūdžio klientams).

2. Kas tai yra *SWOT matrica*? (2)

Tai tam tikru būdu pateikiama SWOT analizės rezultatų sąranka, pagal kuria yra patogu formuoti strategiją. Išskiriamos 4 tipų strategijos:

- „**stiprybė-galimybė" –** numato naudoti galimybes gerai palaikomas verslo stipriųjų pusių;
- **„silpnybė-galimybė" –** galimybes, prieš tai pašalinus silpnąsias verslo puses, trukdančias sėkmingai pasinaudoti tomis galimybėmis;
- **„stiprybė-grėsmė"** – verslas galimybės pasinaudotis stipriosiomis pusėmis išorinėms grėsmėms pašalinti;
- **„silpnybė-grėsmė"** – gynybinius veiksmus, padedančius išvengti dėl verslo silpnųjų pusių kylančių išorinių grėsmių (problemų).

3. Kokias *korporacijos lygmens strategijas* žinote? Trumpai jas apibūdinkite. (4)

- Verslo augimo strategijos:
  - ***Sutelkties*** – verslas sutelkiamas pagrindinių gaminių*(paslaugų)* šeimai ir ieškoma augimo būdų, didinant šios gamybos apimtis;
  - ***Vertikalaus integravimo*** – perimama įeigos ir/arba išeigos logistika;
  - ***Horizontalaus integravimo*** – perimamos varžovų įmonės;
  - ***Plėtros*** – koncentrinė/konglomeratinė plėtra;
  - ***Verslo internacionalizavimo*** – decentralizuotas/globalizuotas/transnacionalinis verslas.
- Stabilumo strategijos – verslo siekis išlaikyti esamą dydį ir esamas operacijų apimtis.
  - ***Stabilus augimas*** – nenumato jokių šuolių, siekiama dirbti kaip visuomet, tik truputį geriau;
  - ***Atsitraukimo*** *(pofit)*.
    - Rušys: ***pasipelnymo*** *(harvest)*; ***žaidimo pabaigos*** *(endgame)*
- Verslo atnaujinimo strategijos – mažėjant pelnui, trūkstant apyvartinių lėšų, klientus perimant kitiems; Galimi du strategijos tipai*(abiems reikia sumažinti kaštus ir restruktūrizuoti verslą)*:
  - *išlaidų mažinimo (retrenchment)* – trumpalaikė, ja siekiama pašalinti silpnąsias verslo puses;
  - *atgaivinimo (turnaround)* – jei situacija besanti rimtesnė – imamasi kardinalesnių priemonių.

5. Kuo ypatingos *viešojo administravimo įstaigų ir pelno nesiekiančių organizacijų strategijos*? (3)

Šių org. veiklos sėkmingumą vertinti sunkiau negu komercinėse organizacijose, nes daug sunkiau sugalvoti tinkamą matų sistemą**.**

- **Tokiose organizacijose vadovybė linkusi galvoti apie didesnius išteklius, bet ne kaip tikslingiau juos naudoti. Dažnai formuluojamos funkcinės strategijos, numatančios padalinių darbo gerinimą** *(organizacijos misijos požiūriu).*
- Nepakankami resursai ir išoriniai ribojimai *(viešųjų pirkimų tvarka, samdymas pagal konkursus ir kt.)* stipriai apriboja strategines galimybes. Tačiau ir viešojo administravimo bei pelno nesiekiančios organizacijos, panašiai kaip komercinės organizacijos, varžosi dėl resursų bei užsakymų.

6. Pagal kokią schemą vyksta *poreikių analizė*? (4)

- ***Pradedant nuo verslo tikslų:*** Išorinė analizė*(problemos, grėsmės, galimybės)*-> Vidinė analizė *(problemų priežastys)*-> Strategija*(kaip pagerinti verslą?)* <- , -> Tikslų medis*(kaip įgyvendinti strategiją?)*-> Operaciniai poreikiai*(kokių IT paslaugų reikia?)*-> Scenarijus*(kaip naudotis sistema?)*-> Įgyvendinimo planas*(ko reikia scenarijui įgyvendinti?)*
- ***Pradedant nuo sistemos:*** Programų sistema *–(įgyvendina)–>* Informacinės ir skaičiavimo paslaugos *(operaciniai poreikiai) –(paremia)–>* Potiksliai *–(sudaro)–>* Tikslų medis *–(įgyvendina)–>* Verslo tobulinimo strategija *–(sprendžia)–>* Verslo problemos

9. Kas tai yra *sistemos naudojimo scenarijus*? Kokia informacija jame pateikiama? Kokiems tikslams jis naudojamas? (4)

Sistemos naudojimo scenarijus aprašo kuriamą sistemą ir jos aplinką, įskaitant ja besinaudojančių agentų elgseną, o taip pat visą kontekstinę informaciją, reikalingą pirminei įgyvendinamumo analizei atlikti.

- SNS aprašo **kaip bus dirbama** organizacijoje po to, kai PS bus sukurta ir įdiegta;
- SNS nusako pačius bendriausius **vartotojo interfeisų reikalavimus** ir sistemos **darbo vietų reikalavimus;**
- SNS aprašomas UML **sekų diagramomis;**
- kas daroma, atliekant rankines operacijas;
- sistemos sąveika su liktinėmis (legacy) PS;
- agentai, jų vaidmenys ir jų darbo aplinka.

10. Kokius veiksmus apima *scenarijaus įgyvendinimo veiksmų planas*? Kokia informacija reikalinga jam parengti? Jo įgyvendinimo schema? (4)

Veiksmai:

- reikiamos TĮ ir PĮ pirkimas;
- kompiuterių tinklo kūrimas;
- personalo mokymas;
- instrukcijų, įsakymų ir kitų reikalingų dokumentų parengimas.

Reikia palyginti esamą padėtį su scenarijuje pateiktais darbo vietų reikalavimais.
Įgyvendinimo schema:

```
Inovaciniai slenksčiai \\
Sistemos naudojimo scenarijus -> Veiksmų planas -> Plano įgyvendinimas -> Naudojama sistema
Esamos situacijos aprašas //
```

1. Kokiais tikslais atliekama *įgyvendinamumo analizė*? Į kokius vadovybei svarbius klausimus ji atsako? (2)

Įgyvendinamumo analizės tikslas yra įsitikinti, kad sistemą galima sukurti ir kurti ją tikrai verta. Analizė turi atsakyti vadovybei į šiuos klausimus:

- Ar projektas įgyvendinamas?
- Kokiais alternatyviais būdais jį galima įgyvendinti?
- Kokiais kriterijais vadovautis parenkant alternatyvą?
- Kuri alternatyva geriausia?

2. Kokie yra *įgyvendinamumo aspektai*? (2)

- **Operacinis** *(ar veiks?)*
- **Techninis** *(ar „išneš“ technologijos?)*
- **Ekonominis** *(ar grįš investicijos?)*
- **Plano** *(ar galima pabaigti laiku?)*
- **Teisinis etinis** *(ar projektas nepažeis kokių nors teisės ar pripažintų etinių normų?)*

3. Kas vadinama *operaciniu įgyvendinamumu*? (3)

Ar užsakovas pajėgus eksploatuoti sistemą? Ar scenarijus iš tiesų veiks? Ar vartotojai suinteresuoti laikytis scenarijumi nustatytų darbo taisyklių? *(pvz., darbo su klaviatūra apimtys, kompiuterių baimė, tradicijos, korporacinė kultūra ir pan.)*
PS naudojamos tam tikrame socialiniame ir organizaciniame kontekste. Tai gali paveikti sistemos reikalavimus *(pvz., interfeiso reikalavimus)* ar netgi lemti projekto sėkmę.

4. Kas vadinama *techniniu įgyvendinamumu*? (4)

- Ar galima turimomis technologijomis sukurti reikiamą sistemą?
- Ar galima sistemą integruoti su kitomis naudojamomis sistemomis?
- Ar vykdytojai pajėgūs sukurti sistemą?
- Ar organizacijos turima*(galima įsigyti)* TĮ tinkama sistemai veikti?
- Ar sistemos kūrimo technologiją vykdytojai jau yra išbandę kurdami kitas panašias sistemas?
- Ar reikės kurti naujus algoritmus ar kokias nors naujas duomenų įvesties arba išvesties technologijas?
- Ar bus naudojama kokia nors nauja, neišbandyta TĮ?
- Ar reikės naudotis kokiais nors neišbandytais komponentais?
- Ar nereikės naudotis vykdytojų neišbandyta DBVS?
- Ar nereikės kurti kokių nors labai specializuotų, neįprastų interfeisų*(pvz., skaityti daviklių rodmenis)*?
- Ar nereikės kurti tokių komponentų, kokių vykdytojai dar nėra kūrę? Kiek komponentų procentais?
- Ar nereikės naudoti anksčiau nenaudotų analizės, projektavimo ar testavimo metodų?
- Ar nebus naudojamos retai naudojamos PS paradigmos*(pvz., sintezės paradigma)* arba kokie nors sudėtingi metodai*(duomenų analizės metodai, dirbtiniai neuroniniai tinklai ar pan.)*?
- Ar sistemai nėra keliami išskirtiniai našumo reikalavimai?
- Ar užsakovai neabejoja operacinių reikalavimų įgyvendinamumu?

5. Kas vadinama *ekonominiu įgyvendinamumu*? Kokia *išlaidų ir pajamų analizės* esmė? (5)

- Daugeliui projektui tai pats svarbiausias įgyvendinamumo aspektas!
- Analizuojama, ar gauti privalumai tikrai pakankami tokioms investicijoms.
- Kiekvienai sistemos reikalavimų alternatyvai vertinami jos kaštai ir privalumai.
- Tai vadinama **išlaidų ir pajamų analize**.

Šios analizės tikslas yra atsakyti į klausimus:

- Ar projektas pateisinamas (pelnas viršys išlaidas)?
- Ar numatomų investicijų pakanka projektui įgyvendinti?
- Už kokią mažiausią sumą galima sukurti priimtiną sistemą?
- Sunkumai – identifikuoti ir įvertinti visas išlaidas ir pajamas

Pajamos ir išlaidos gali būti netiesioginės ar sunkiai įvertinamos, dažnai sunku vertinti daugeliu kriterijų vertinamas alternatyvas.

6. Kokios *pajamų ir išlaidų rūšys* yra analizuojamos išlaidų ir pajamų analizės metu? (4)

Galima tokia iš sistemos gaunama nauda:

- ***Piniginė*** – galima apskaičiuoti pinigine išraiška;
- ***Įvertinama*** – galima įvertinti, bet ne pinigais;
- ***Neapčiuopiama*** – negalima įvertinti nei vienu, nei kitu būdu.

Pajamų*(naudos)* rūšys:

- Naujos ar tobulesnės galimybės;
- Efektyvesnės operacijos ir didesnis tikslumas;
- Greitesnis sprendimų priėmimas;
- Išlaidų taupymas!

Išlaidų rūšys:

- **TĮ,** įskaitant serverius;
- **PĮ –** sistemai kurti, aptarnauti, dokumentuoti, personalui mokyti ir t.t.;
- **Eksploatavimo kaštai –** priežiūra ir t.t.;
- **Papildomas personalas –** sistemai administruoti, duomenims ruošti ir t.t.;
- **Diegimo išlaidos;**
- **Pradinės investicijos.**

7. Kokios *veiklos* vykdomos atliekant poreikių ir išlaidų/pajamų analizes? (3)

- Verslo problemų ir grėsmių nustatymas ir išsiaiškinimas;
- Alternatyvių verslo tobulinimo strategijų formulavimas;
- Sistemos naudojimo scenarijų sudarymas (bent po vieną kiekvienai strateginei alternatyvai);
- Strategijos alternatyvų (ir atitinkamų naudojimo scenarijų) analizė pagal šiuos kriterijus:
    - resursų poreikis, kūrimo ir eksploatavimo kaštai, projekto trukmė.
- Alternatyvių (tai pačiai strategijai) sistemos naudojimo scenarijų vertinimas:
    - išlaidų ir pajamų analizė kiekvienam scenarijui;
    - investicijų grąžos skaičiavimas;
    - alternatyvų ir išlaidų/pajamų sugretinimo lentelė;
    - Geriausios strategijos ir geriausio scenarijaus parinkimas.
- Gali paaiškėti, kad nei vieno iš siūlomų scenarijų negalima įgyvendinti dėl:
    - per didelių inovacinių slenksčių ar per didelių kaštų;
    - apribojimų resursams ar techninių priežasčių;
    - juridinių ribojimų.

8. Kas tai yra *konteksto analizė*?

Konteksto analizė atliekama tikslu atsakyti į šiuos klausimus:

- Kaip PS padės verslui?
- Kokias užduotis ir kokiu mastu ji padės vykdyti?
- Kokią konkrečią naudą iš to gaus agentai?