Architektūros projektavimas

Pagal I.Sommerville "Software Engineering", 9 leidimo 6 dalį

1

Nagrinėjamos temos

- ✧ Architektūrinio projektavimo sprendimai
- ✧ Požiūris į architektūrą
- ✧ Architektūros šablonai
- ✧ Programų architektūra

2

PĮ architektūra

- Tai projektavimo procesas, kurio metu identifikuojamos sistemos posistemės, tų posistemių kontrolės ir tarpusavio sąveikos būdai.
- Rezultatas – PĮ architektūros dokumentas.

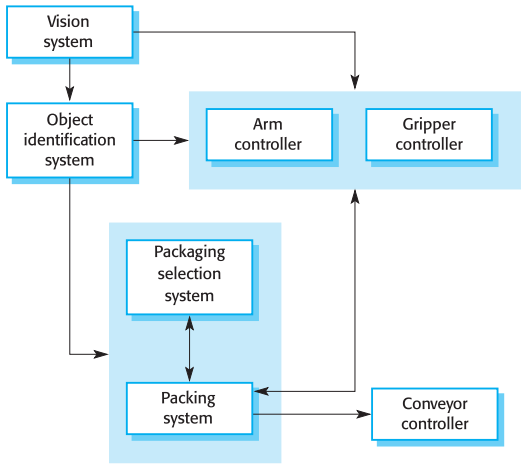
3

PĮ architektūra

- Tai pradinis sistemos projektavimo darbas.
- Atspindi sąryšį tarp specifikacijos ir projektavimo proceso.
- Dažnai ruošiamas lygiagrečiai su sistemos specifikacijos dokumentu.
- Apima pagrindinių sistemos komponentų bei jų tarpusavio sąryšio nustatymą.

4

Roboto, kuris pakuoja prekes, sistemos architektūra



5

Architektūros sąvokos

- Smulkioji architektūra: apima konkrečios vienos programos architektūrą, nagrinėjama, kaip ta programa skaidoma į komponentus.
- Stambioji architektūra: apima kelias programas, sudarančias vieną sudėtingą sistemą. Tokia sistema gali būti parašyta ne vienos įmonės bei veikti ne tik viename kompiuteryje, bet dideliame kompiuterių tinkle.

6

Architektūros dokumento privalumai

- ✧ Suinteresuotų asmenų bendravimas
  - Architektūros dokumentas gali būti naudojamas diskusijose su užsakovu.
- ✧ Sistemos analizė
  - Galima patikrinti, ar sistema tenkins jai iškeltus nefunkcinius reikalavimus.
- ✧ Pakartotinis naudojimas
  - Architektūra gali būti naudojama pakartotinai
  - Galima kurti produktų-linijos architektūras.

7

Architektūros vaizdavimas

- ✧ Dažniausiai naudojamos paprastos blokinės diagramos, parodančios visas esybes ir sąryšius.
- ✧ Tačiau tokias diagramas kartais kritikuoja dėl to, jog jos nerodo esybių tipų ar savybių, taip pat nerodomi ryšiai tarp tų esybių.
- ✧ Priklauso nuo to, koks architektūrinis modelis pasirenkamas.

8

Blokinės ir linijinės diagramos

- ✧ Labai abstrakčios – nerodo komponentų sąryšio, nei sistemos savybių.
- ✧ Naudingos bendraujant su klientu ir planuojant projektą
  - Klientas supranta paprastas diagramas
  - Projekto vadovas jau žino, kokie komponentai bus kuriami, ir kokių žmonių jam reikės, todėl gali pradėti planuoti resursus.

9

Architektūros projektavimo metu priimami sprendimai

- ✧ Tai kūrybinis procesas, todėl visi procesai gali būti skirtingi priklausomai nuo to, kokią programą mes kuriame..
- ✧ Tačiau galima išskirti keletą bendrų klausimų, kurie yra sprendžiami tam, kad gerai suprojektuotume nefunkcinių sistemos reikalavimų įvykdymą.

10

Architektūros projektavimo metu kylantys klausimai

- ✧ Ar turime kokią nors bendro pobūdžio architektūrą, kurią galėtume dabar panaudoti?
- ✧ Kaip bus paskirstyta sistema?
- ✧ Kokie architektūros stiliai šiuo atveju yra tinkami?
- ✧ Kokį metodą naudosime sistemos struktūros sudarymui?
- ✧ Kaip sistemą skaidysime į modulius?
- ✧ Kokią kontrolės sistemą naudosime?
- ✧ Kaip mes realizuosime tą architektūros projektą, kurį ką tik sukūrėme?
- ✧ Kaip dokumentuosime architektūrą?

11

Pakartotinis architektūros projektavimas

- ✧ Tos pačios srities sistemos dažnai turi panašią architektūrą, atspindinčią tos konkrečios srities savybes.
- ✧ Pradžiai kuriamas bendras architektūros modelis, kuris po to pritaikomas konkrečiam klientui pagal jo poreikius.
- ✧ Architektūra gali būti projektuojama laikantis kurio nors vieno šablono ar stiliaus.
  - Aptarsime vėliau.

12

Architektūra ir sistemos charakteristikos

- Našumas (Performance)
  - Nustatomos kritinės operacijos ir mažinama komunikacija tarp tas operacijas atliekančių komponentų. Dažniau naudojami stambūs, negu smulkūs komponentai.
- Saugumas (Security)
  - Naudojama sluoksninė (layered) architektūra, kur svarbiausi dalykai yra prieinami tik žemiausiame lygyje ir ne visiems vartotojams.
- Saugumas (Safety)
  - Visos su saugumu susijusios savybės dedamos į kuo mažesnį skaičių modulių, nes taip lengviau jas suvaldyti.
- Prieinamumas(Availability)
  - Sistema projektuojama taip, kad tam tikras jos dalis galima būtų pakeisti ir atnaujinti nestabdant visos sistemos. Būtina turėti gerą klaidų apdorojimą.
- Palaikomumas (Maintainability)
  - Naudojamus komponentus galima nesunkiai pakeisti kitais.

13

Požiūris į architektūrą

- Iš kokios perspektyvos mums reikia žiūrėti, kai dokumentuojame ir projektuojame sistemos architektūrą?
- Kokias sąvokas turėtume naudoti sistemos architektūros modelių aprašymui?
- Kiekvienas architektūrinis modelis rodo sistemos savybes tik iš kurios nors vienos perspektyvos.
  - Pvz. Gali rodyti, kaip sistema yra išskaidyta į modulius, kaip siejasi realaus laiko procesai tarpusavyje, jei sistema yra paskirstyta tinkle.
  - Paprastai reikia kelių modelių, parodančių sistemą iš skirtingų perspektyvų.

14

4 + 1 požiūris į sistemos architektūrą

- Loginis požiūris, pagrindines sistemos abstrakcijas rodantis kaip objektus ar objektų klases.
  - Turi būti galima susieti sistemos reikalavimus su objektais iš šio loginio požiūrio.
- Procesų požiūris, kuris rodo, kaip veikimo metu sistema yra skaidoma į sąveikaujančius procesus.
  - Naudinga, kai reikia kontroliuoti nefunkcinių reikalavimų vykdymą
- Kūrimo požiūris, kuris rodo, kaip sistema yra išskaidoma jos kūrimo (programavimo) atžvilgiu.
  - Galima iš anksto žinoti, kokie konkrečiai moduliai bus skiriami vienai ar kitai programuotojų komandai.
- Fizinis modelis, rodantis kompiuterinės ir programinės įrangos komponentų skirstymą sistemos procesoriams.
- Vartotojo scenarijai ir panaudos atvejų diagramos (+1)

15

Architektūros šablonai (patterns)

- Šablonai skirti atvaizduoti, dalintis ir pakartotinai naudoti tam tikras žinias.
- Architektūros šablonas yra stilizuotas geros projektavimo praktikos, išbandytos kuriant ne vieną skirtingą sistemą skirtingoje aplinkoje, aprašymas.
- Šablonai turi turėti informaciją kada jie TURI būti naudojami, ir kada – NE.
- Šablonai gali būti vaizduojami ir grafiniu, ir tekstiniu būdu.

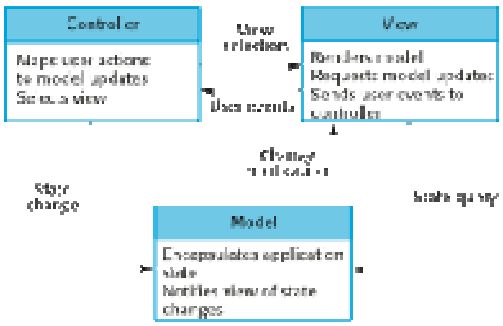
16

Model-View-Controller (MVC) šablonas

Pavadinimas	MVC (Model-View-Controller)
Aprašymas	Atskiria vaizdą ir sąveiką nuo sistemos duomenų. Sistema skaidoma į tris loginius komponentus, sąveikaujančius tarpusavyje. **Modelio** komponentas valdo sistemos duomenis ir su jais susijusias operacijas. **Vaizdo** (View) komponentas nusako, kaip duomenys bus rodomi vartotojui. **Kontroliavimo** (Controller) komponentas valdo vartotojo veiksmus (pvz. Pelės paspaudimas, klavišo paspaudimas ir pan.) ir siunčia šiuos veiksmus į Vaizdo ir Modeliavimo komponentus. Žr. Pav. Kitoje skaidrėje
Pavyzdys	Kitoje skaidrėje parodyta webinės aplikacijos architektūra, suprojektuota pagal MVC šabloną.
Kada naudojamas	Naudojamas tada, kai galima keletu skirtingų būdų peržiūrėti ir pasiekti reikiamus duomenis. Taip pat naudojamas tada, kai nežinomi būsimi reikalavimai duomenų peržiūrėjimui ir pasiekimui.
Privalumai	Leidžia keisti duomenis nepriklausomai nuo jų vaizdavimo būdo ir atvirkščiai. Palaiko tų pačių duomenų vaizdavimą skirtingais būdais keičiant duomenis tik vienoje vietoje.
Trūkumai	Gali reikėti sudėtingo programos kodo net ir tada, kai duomenų modelis bei sąveika tarp duomenų yra labai paprasti.

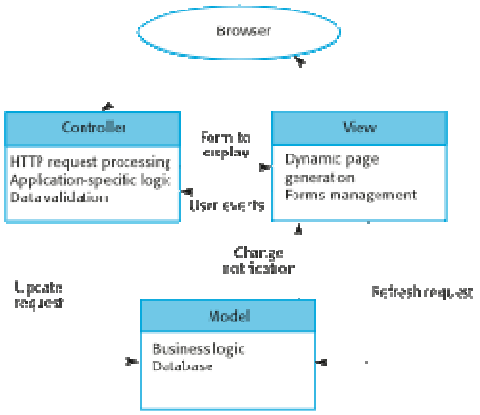
17

Model-View-Controller schema



18

Internetinės svetainės architektūra naudojant MVC šabloną



19

Sluoksnių (layered) architektūra

- ✧ Naudojama posistemių tarpusavio sąsajoms modeliuoti.
- ✧ Sistemą suskirsto į tam tikrus lygius, iš kurių kiekvienas atlieka vis kitas funkcijas.
- ✧ Palaiko palaipsninį sistemos kūrimą, kai keičiama vieno lygio sąsaja, tai paveikiamas tik vienas sluoksnis.

20

Sluoksninės architektūros šablonas

Pavadinimas	Sluoksninė architektūra
Aprašymas	Suskirsto sistemą į sluoksnius pagal panašų funkcionalumą, būdingą kiekvienam sluoksniui. Sluoksnis teikia paslaugas aukštesniam sluoksniui, todėl žemiausias sluoksnis apima žemiausio lygio paslaugas, reikalingas visiems sluoksniams.
Pavyzdys	Sistemos, leidžiančios dalintis autoriniais dokumentais, laikomais skirtingose bibliotekose, sluoksninis modelis pavaizduotas 22 skaidrėje.
Kada naudojamas	Naudojamas tada, kai naujos sistemos kuriamos ant senos sitemos pagrindo; kai sistemos kūrimas padalintas skirtingoms programuotojų komandoms prieskiriant vienai komandai vieną sluoksnį; kai reikalaujama keleto sugumo lygių.
Privalumai	Galima vidinius sluoksnius keisti daug kartų tol, kol palaikoma sluoksnio sąsaja (interfeisas). Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.
Trūkumai	Praktikoje atskirti vieną sluoksnį nuo kito yra gan sunku, todėl dažnai aukštesnis sluoksnis sąveikauja tiesiai su žemiausio sluoksnio elementais, nes tarpinį sluoksnį sukurti toje konkrečioje situacijoje būna neįmanoma. Gali būti našumo problemų.

21

Sluoksninės architektūros šablonas

Vartotojo sąsaja

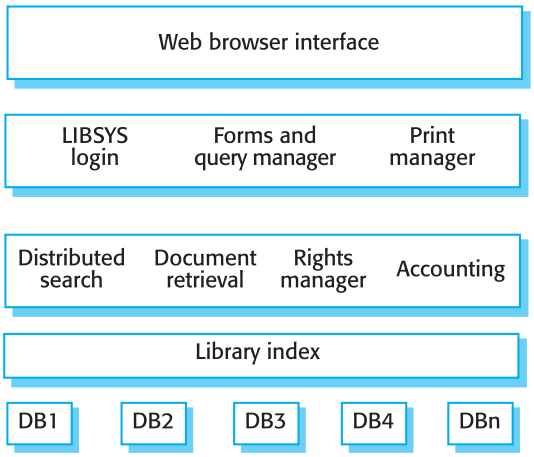
Vartotojo sąsajos valdymas, autorizacija ir identifikacija

Bazinė verslo logika/sistemos funkcionalumas
Sistemos teikiamos naudos

Sistemos palaikymas(OS, duomenų bazė ir pan.)

22

Bibliotekos sistemos LIBSYS architektūra pagal sluoksninį šabloną



23

Saugyklos (Repository) architektūra

- Posistemės turi keistis duomenimis. Tas gali būti daroma dviem būdais:
  - Bendri duomenys laikomi centrinėje duomenų bazėje ir yra prieinami visoms posistemėms;
  - Kiekviena posistemė turi savo duomenų bazę ir duomenis perduoda ne per duomenų bazę, bet tiesiogiai.
- Kai reikia dalintis dideliu kiekiu duomenų, naudojamas būtent šis modelis.

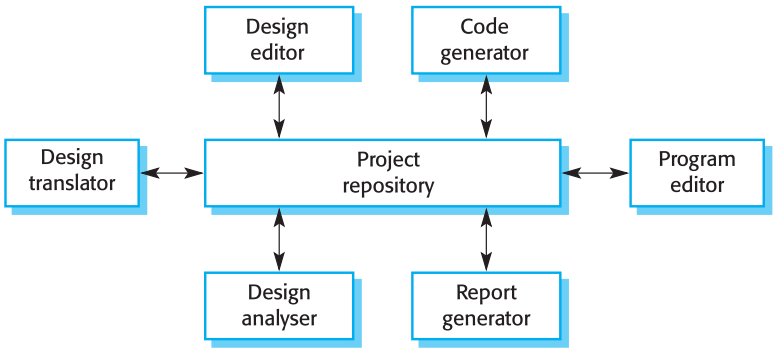
24

Saugyklos šablonas

Pavadinimas	Saugykla
Aprašymas	Visi sistemos duomenys valdomi centrinėje saugykloje, visi sistemos komponentai gali prieiti prie tos saugyklos. Komponentai tiesiogiai duomenimis nesikeičia, duomenų apsikeitimas galimas tik per saugyklą.
Pavyzdys	Skaidė 25, joje pavaizduota sistema, kurios komponentai naudojami sistemos kūrime. Kiekvienas programavimo įrankis sugeneruoja informaciją, kuri vėliau prieinama ir kitiems programavimo įrankiams.
Kada naudojama	Šį modelį siūloma naudoti tada, kai turite sistemą su dideliu informacijos kiekiu, kurį reikia saugoti ilgą laiką. Taip pat šį modelį galima naudoti tada, kai duomenų padėjimas į saugyklą iššaukia tam tikrą įvykį ar įrankio naudojimą.
Privalumai	Komponentai yra nepriklausomi – jiems nereikia žinoti apie vienas kito egzistavimą. Pakeitimai, padaryti viename sistemos komponente, gali būti automatiškai perduodami kitiems komponentams. Duomenys valdomi nuosekliai, nes jie yra visi vienoje vietoje (pvz. atsarginė kopija padaroma visiems komponentams vienu metu).
Trūkumai	Jei kas nors nutinka saugyklai, tai tas gedimas paveiks visą sistemą. Gali būti, kad nepakankamai efektyviai dalinamasi duomenimis. Saugyklos paskirstymas po atskirus kompiuterius gali būti sudėtingas.

25

Saugyklos architektūros pavyzdys – CASE įrankių sistema



26

Kliento-serverio architektūra

- Tai paskirstytų sistemų modelis, rodantis kaip tarp skirtingų komponentų paskirstomi duomenys ir procesai.
  - Galima pritaikyti ir vienam kompiuteriui.
- Aibė atskirų serverių, atliekančių specifines užduotis, pvz. Spausdinimas, duomenų valdymas ir pan.
- Aibė klientų, kurie naudojasi šiomis paslaugomis.
- Tinklas, leidžiantis klientams prieiti prie reikiamų paslaugų.

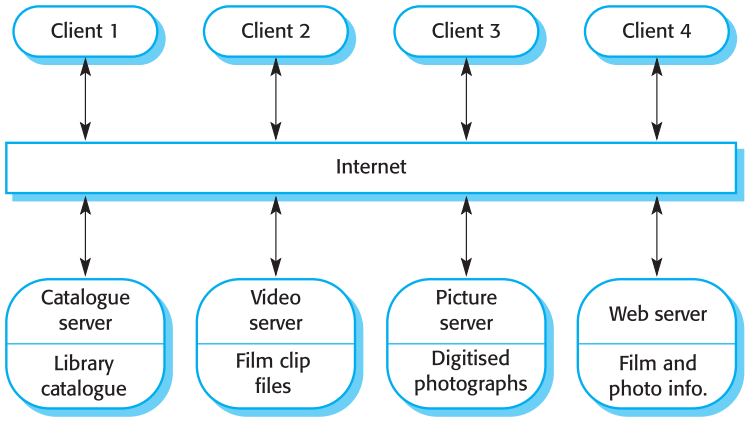
27

Kliento-serverio šablonas

Pavadinimas	Client-server
Aprašymas	Kliento-serverio architektūroje sistemos funkcionalumas organizuojamas per paslaugas (servise), gaunamus iš skirtingų serverių. Klientai yra šių paslaugų vartotojai ir jungiasi prie serverio tam, kad gautų tas reikiamas paslaugas.
Pavyzdys	Skaidrė 29, filmų ir video/DVD bibliotekos pavyzdys.
Kada naudojamas	Naudojamas tada, kai visiems prieinamoje duomenų bazėje esantys duomenys turi būti pasiekiami iš skirtingų vietų. Taip pat naudojama tada, kai sistemos apkrovimas yra nepastovus.
Privalumai	Pagrindinis privalumas – serveriai gali būti paskirstyti tinkle. Bendras funkcionalumas, pvz. spausdinimas, gali būti prieinamas visiems klientams ir nėra reikalo tą funkcionalumą diegti į visas paslaugas.
Trūkumai	Sugedus serveriui, atskiros paslaugos taip pat nebus teikiamos. Našumas gali būti nepakankamas, nes visa sistema priklausi nuo tinklo ir interneto greičio. Gali būti valdymo problemų, jeigu serveriai priklauso skirtingoms organizacijoms.

28

Kliento-serverio architektūra filmų bibliotekos sistemai



29

Programų architektūra

- Sistemos kuriamos tam tikrų verslo tikslų įvykdymui.
- Jei keli verslai yra panašūs, tai ir sistemos greičiausiai turės bendrų architektūros elementų, nusakančių tos programos reikalavimus.
- Bendro pobūdžio programų architektūra – tai tokia architektūra, kuri gali būti pritaikoma sistemos su konkrečiais reikalavimais kūrimui.

30

Programų architektūros naudojimas

- Pradinis taškas sistemos projektavimui.
- Pagalbinė priemonė, leidžianti pasitikrinti, ar ko nors nepamiršome.
- Vienas iš būdų paskirstyti darbus programuotojams.
- Padeda indentifikuoti, kuriuos komponentus galėsime naudoti pakartotinai.
- Galima naudoti vietoj terminų žodyno tada, kai kalba eina apie sistemos tipus.

31

Sistemų tipų pavyzdžiai

- Duomenų perdavimo programos
  - Programos, kurios perduoda duomenis be vartotojo įsikišimo į procesą.
- Tranzakcijų (transaction processing) apdorojimo programos
  - Į duomenis orientuotos programos, kurios apdoroja vartotojų užklausas ir atnaujina duomenis sistemos duomenų bazėje.
- Įvykių apdorojimo sistemos
  - Tai programos, kuriose veiksmai priklauso nuo sistemos aplinkos interpretavimo.
- Kalbos (Language) apdorojimo sistemos
  - Tai programos, kuriose vartotojo veiksmai nusakomi tam tikromis komandomis (žodžiais), kurias sistema supranta ir gali apdoroti.

32

Sistemų tipų pavyzdžiai

- Panagrinėkime tranzakcijų ir kalbos apdorojimo sistemas.
- Tranzakcijų apdorojimo sistema
  - E-komercijos sistemos;
  - Rezervavimo sistemos.
- Kalbos apdorojimo sistemos
  - Kompiliatoriai;
  - Komandų interpretatoriai.

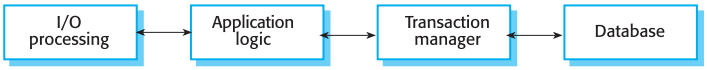
33

Tranzakcijų apdorojimo sistemos

- Apdoroja vartotojo užklausas gauti tam tikrą informaciją iš duomenų bazės.
- Iš vartotojo perspektyvos, tranzakcija yra:
  - Bet kokia operacijų seka, leidžianti pasiekti tikslą;
  - Pvz. – rasti lėktuvų skrydžių iš Londono į Paryžių tvarkaraščius.
- Vartotojai užklausas įveda ne vienu metu, o užklausų valdymo mechanizmas jas apdoroja.

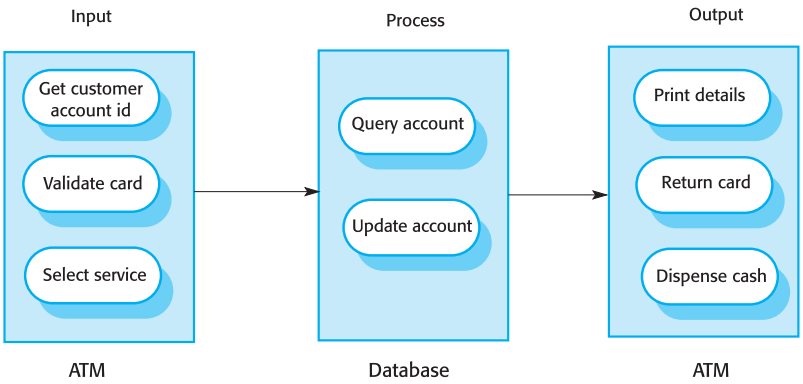
34

Tranzakcijų apdorojimo programos struktūra



35

Bankomato sistemos architektūra



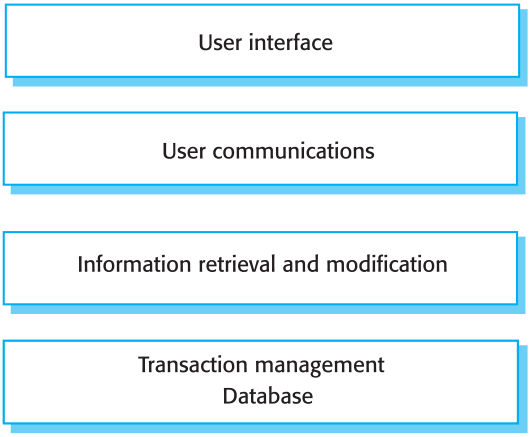
36

Informacinių sistemų architektūra

- Informacinės sistemos turi bendro pobūdžio architektūrą, kuri gali būti projektuojama sluoksniais.
- Tokios sistemos priskiriamos tranzakcijų apdorojimo sistemoms, nes tokiose sistemose dažniausiai yra būtina turėti duomenų bazę.
- Sluoksniai apima:
  - Vartotojo sąsają
  - Vartotojo bendravimą
  - Informacijos gavimą
  - Sistemos duomenų bazę

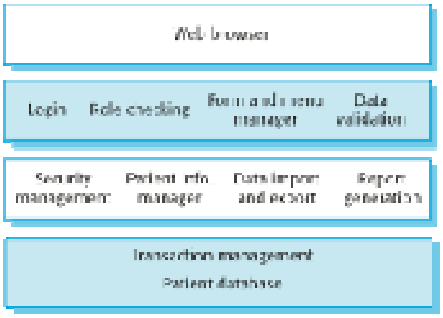
37

Informacinių sistemų architektūros sluoksniai



38

Ligoninės sistemos MHC-PMS architektūra



39

Internetinės informacinės sistemos

- Informacinės sistemos šiuo metu dažniausiai veikai internete, kur vartotojo sąsaja yra pasiekiama naudojant tam tikrą naršyklę.
- Pvz. E-prekybos sistemos priima užsakymus prekėms ar paslaugoms, sudaro tų prekių ar paslaugų pristatymo klientui tvarkaraščius.
- E-prekybos sistemose reikia papildomo sluoksnio “prekių krepšelis”, kur vartotojas atskiromis tranzakcijomis sudeda prekes, o užmoka už visas prekes viena tranzakcija.

40

Serverio naudojimas

- Web serveris atsakingas už visą vartotojo komunikaciją su interneto naršykle.
- Serveris atsakingas už specifinės logikos realizavimą saugant arba gaunant reikiamą informaciją.
- Serveris sudeda ir paima reikiamą informaciją į/iš duomenų bazę.

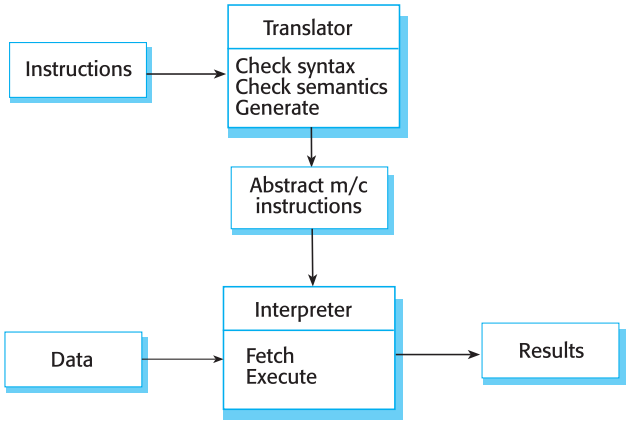
41

Kalbos apdorojimo sistemos

- Priima natūralią ar dirbtinę kalbą ir ją apdoroja.
- Gali turėti kalbos kalbos interpretatorių.
- Naudojamas tada, kaip lengviausias būdas išspręsti problemą tai aprašyti algoritmą ar sistemos duomenis

42

Kalbos apdorojimo sistemos architektūra



43

Kompiliatoriaus elementai

- Kalbos analizatoriai, kurie priima kalbą ir išverčia ją į tam tikrą vidinį formatą.
- Simbolių lentelė, kurioje surašyta informacija apie esybių vardus (kintamieji, klasių vardai, objektų vardai), naudojamus šifruojant tą įvestą kalbą.
- Sintaksės analizė, tikrinanti, ar mes gerai įvedėme tekstą.
- Sintaksės medis, kuris nusako vidinę programos, kurią mes kompiliuojame, struktūrą.

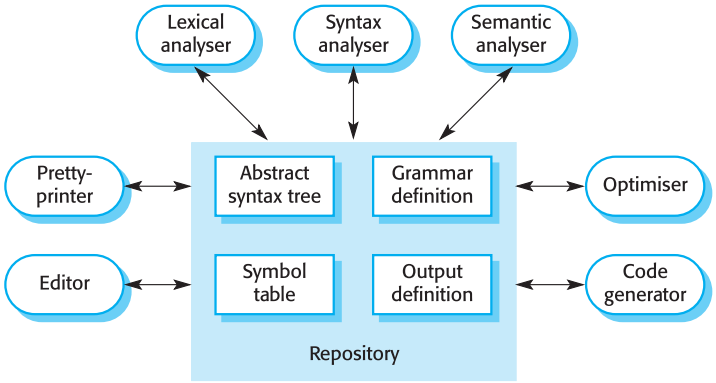
44

Kompiliatoriaus komponentai

- Semantinis analizatorius, kuris naudoja sintaksės medžio ir sintaksės lentelės informaciją tam, kad patikrintų semantinį įvestos kalbos teisingumą.
- Kodo generatorius, kuris eina per sintaksės medį ir sudaro abstraktų mašininį kodą.

45

Saugyklos tipo architektūra kompiliatoriaus sistemai



46

Klausimai

47