[Objektno usmjerene arhitekture]

Klijent – poslužitelj arhitektura

Prednosti
- Posao se može **raspodijeliti** na više računala (strojeva).
- Klijenti **udaljeno** pristupaju funkcionalnostima poslužitelja.
- Klijent i poslužitelj mogu se **oblikovati odvojeno**.
- Oba entiteta mogu biti **jednostavnija**.
- Svi podaci mogu se držati **na jednom mjestu** (na poslužitelju).
- Obrnuto, podaci se mogu **distribuirati** na više udaljenih klijenata i poslužitelja.
- Poslužitelju mogu **simultano** pristupiti više klijenata.
- Klijenti mogu ući u **natjecanje** (kompeticiju) za uslugu poslužitelja (a i obrnuto).

Tipovi ponovnog korištenja (engl. ***reuse*****):**
- Ponovno koristi ekspertizu (posebna znanja).
- Ponovno koristi standardna oblikovanja i algoritme.
- Ponovno koristi knjižnice razreda ili procedura.
- Ponovno koristi snažne naredbe ugrađene u programski jezik ili operacijski sustav.
- Ponovno koristi **radne okvire (**engl. **frameworks).**
- Ponovno koristi cijela primjenska rješenja (aplikacije).

Arhitektura s više razina (engl. n-tier)

- **Prednosti:**
  - Oblikovanje temeljem više razine apstrakcije.
  - Podupire povećanje i poboljšanje sustava (promjena na jednoj razini utječe samo još na razinu ispod i iznad).
  - Podupire ponovno korištenje, prenosivost i sl.

- **Nedostaci:**
  - Teško je odrediti optimalno preslikavanje odgovornosti na razine.
  - Ponekad se izračunavanje i funkcionalnosti sustava ne mogu razbiti na razine.
  - Ako se želi poboljšati performanse mora se preskakati ili "tunelirati" kroz razinu.

Arhitektura zasnovana na komponentama

CORBA
Dobro: standard (OMG grupa), transparentna komunikacija između objekata koji su oblikovani različitim programskim jezicima i žive na različitim strojevima, u heterogenoj raspodijeljenoj okolini.
Loše: Definirani su na razini izvornog koda, a ne na binarnoj razini. To vrlo usporava rad jer se komunikacija odvija na visokoj razini definiranih protokola. Svi programski jezici moraju imati kopče za CORBA sučelje.

Java/JavaBeans:
Dobro: neovisnost o radnoj platformi, kao reakcija na događaj *Beans* komponenta može komunicirati i spajati se s ostalim *Beans* komponentama, *Beans* komponenta se može prilagođavati specijalnoj primjeni, dostupan izvorni kod.
Loše: pretpostavka virtualnog stroja usporava rad, otkriva se unutarnja struktura.

.NET
Dobro: binarni i mrežni standard za komunikaciju između objekata, primjena u više programskih jezika (C#, VB, Javascript, VisualC++) , moguća je implementacija više globalno poznatih sučelja (prva metoda u sučelju je **queryInterface()** , vraća oznaku ako sučelje nije podržano).
Loše: upravljanje memorijom, kompatibilnost (MSWindows).

[Arhitekture protoka podataka]

Razlozi izbora arhitekture protoka podataka:

- Zadatkom dominira dobavljivost podataka.
- Podaci se mogu prediktivno prenositi od procesa do procesa.
- Cjevovodi i filtri su dobar izbor u mnogim primjenama protoka podataka jer:
    - Dozvoljavaju ponovnu uporabu i rekonfiguraciju filtara.
    - Rasuđivanje o cijelom sustavu je olakšano.
    - Reducira se ispitivanje (testiranje) sustava.
    - Može se ostvariti inkrementalna i paralelna obrada.

- Arhitektura protoka podataka može reducirati performanse sustava (npr. svođenjem na jedan tip podataka u cjevovodima - UNIX).

Osnovna prednost ove arhitekture:
Razdvajanje procesnih dijelova programa i minimizacija dijelova programa koji se odnose na eksplicitno povezivanje varijabli, funkcija, modula, … **.**

[Ostale Arhitekture]

ARHITEKTURA ZASNOVANA NA DOGAĐAJIMA

Prednosti:
- Omogućuje razdvajanje i autonomiju komponenata.
- Snažno podupire evoluciju i ponovno korištenje.
- Jednostavno se uključuju nove komponente bez utjecaja na postojeće.

Nedostaci:
- Komponente koje objavljuju događaje nemaju garancije da će dobiti odziv.
- Komponente koje objavljuju događaje nemaju utjecaja na redoslijed odziva.
- Apstrakcija događaja ne vodi prirodno na postupak razmjene podataka (možda je potrebno uvođenje globalnih varijabli).

- Teško rasuđivanje o ponašanju komponenata koje objavljuju događaje i pridruženim komponentama koje su registrirane uz te događaje (nedeterministički odziv)

ARHITEKTURA REPOZITORIJA PODATAKA

Stil: Oglasna ploča (*engl. blackboard*)

Intencijsko i reaktivno ponašanje – kako se napreduje prema rješenju

Prednost: jednostavno integriranje različitih autonomnih sustava.

Nedostatak: Oglasna ploča može biti vrlo složena arhitektura, posebice ako su komponente prirodno međuzavisne.

Predstavljanje i obradba neizvjesnog znanja

Prednost: Oglasna ploča je dobra arhitektura za ovaj tip problema.

Sigurnost i tolerancija na kvarove

Prednost: podsustavi mogu promatrati oglasnu ploču i obratiti pažnju na potencijalne poteškoće.

Nedostatak: Oglasna ploča je kritičan resurs.

Fleksibilnost

Prednost: oglasna ploča je inherentno fleksibilna

SLOJEVITA ARHITEKTURA PROGRAMSKE POTPORE

Prednosti slojevite arhitekture:

- Sloj djeluje kao koherentna cjelina.
- Vrlo jednostavna zamjena sloja novijom inačicom.
- Jednostavno održavanje jer svaki sloj ima dobro definiranu ulogu.
- Minimizirana je međuovisnost komponenata.
- Podupire oblikovanje programske potpore na visokoj apstraktnoj razini.
- Podupire ponovno korištenje (*engl, reuse*).

Nedostaci slojevite arhitekture:

- Smanjene performanse jer gornji slojevi samo indirektno dohvaćaju donje slojeve.
- Teško se pronalazi ispravna apstrakcija (posebice u sustavima s većim brojem slojeva).
- Svi sustavi se ne preslikavaju izravno u slojevitu arhitekturu (djelokrug programske jedinice može zahvaćati više slojeva).