Rendszerterv

 Funkcionális terv
    ⊗ Rendszerszereplők
    ⊗ Rendszerhasználati esetek és lefutásaik
    ⊗ Menü-hierarchiák
    ⊗ Képernyőtervek
 Fizikai környezet
    ⊗ Vásárolt szoftverkomponensek és külső rendszerek
    ⊗ Hardver és hálózati topológia
    ⊗ Fejlesztő eszközök
 Architekturális terv
 Adatbázis terv
 Implementációs terv
    ⊗ Perzisztencia-osztályok
    ⊗ Üzleti logika osztályai
    ⊗ Kliensoldal osztályai

A tervezési folyamatban célszerű az általános logikai felépítést mindig valamilyen grafikus reprezentációval modellezni, mert az emberi agy vizualizált objektumokkal könnyebben dolgozik.
Az **UML** (=Unified Modeling Language), modellező nyelv segítségével a specifikációt és a tervezést is grafikus formában, diagramok segítségével tudjuk dokumentálni. A követelményspecifikációban a **használati esetdiagram**okat (use case), az adatbázis tervezésnél az **adatbázis diagram**ot, az OOP tervezésnél az **osztálydiagram**ot és **objektumdiagram**ot valamint a **szekvenciadiagram**ot, **aktivitásdiagram**ot használhatjuk.

Architekturális terv

A rendszer jól elkülöníthető feladatai:
- adatok tárolása
- adatokat beolvasása,
- adatok feldolgozása
- eredmények megjelenítése

Monolit rendszer

Az egész rendszer egy nagy program, illetve az adatokat valamilyen relációs adatbázisban tároljuk (pl. MySql, Access), ami a programtól külön fut, de ugyanazon a számítógépen.

Elosztott rendszerek

Az információs rendszer feladatait nem egyetlen nagy gép valósítja meg. A nagy gép (szerver) végzi a tárolást, elvégez egyes feladatokat, az asztali gépeken (kliens) pedig olyan alkalmazások futnak, amelyek a nagy géptől adatokat és/vagy munkavégzést (szolgáltatás - service) igényelnek. A köztük lezajló információ áramláshoz hálózatra van szükség.

Két szintű kliens/szerver architektúra

 A szerveren tárolódnak az adatok, és esetleg ott történik az adatfeldolgozás.
 A kliensen történik a beolvasás, és a megjelenítés, erősebb kliensek esetében az adatfeldolgozás is.
    ⊗ Vékony kliens esetében csak az adat beolvasás és a megjelenítés történik helyben. A szerver végzi az adatszolgáltatást, és az adatok feldolgozását.
    ⊗ Vastag kliens esetében az adat beolvasás, a feldolgozás, és a megjelenítés is helyben történik. A szerver csak adatszolgáltatást végez.

Háromszintű kliens/szerver architektúra

(1) Megjelenítési réteg: biztosítja az alkalmazás felhasználói felületét, vagyis a felhasználói beavatkozások hatására meghívja a megfelelő üzleti logikai funkciót, majd a hívás eredményének megfelelően frissíti a felhasználói felületet.

(2) Alkalmazás réteg: valójában ez egy köztes réteg az adatbázis és a megjelenítési réteg között. A konkrét alkalmazási terület igényeinek megfelelő funkcionalitást biztosítja oly módon, hogy az üzleti szabályok figyelembevételével hívja meg az adatréteg szolgáltatásait. Ezt a réteget ***üzleti logikai rétegnek*** is nevezik.

(3) Adatréteg: A legalsó réteg, melynek feladata az adatok perzisztens tárolása, és az azokon végezhető elemi műveletek támogatása. Leggyakrabban ezt a réteget relációs adatbázis segítségével valósítják meg. Nem ritka a teljesítmény és biztonság növelés céljából kialakított elosztott adatbázisok alkalmazása sem.

A rétegek kialakításával jól elválnak egymástól az egyes funkciók. Könnyebben lehet az egyes funkciókat megvalósító programokat egy másikkal lecserélni. Például egy olcsóbb adatbázis szerverre történő áttérés. Vagy épp egy drágábbra, de jóval gyorsabbra. Ugyancsak előnyös lehet, hogy az adatbázis (és az adatbázis szerver) fizikailag lehet egész máshol, ahol rendszeresen történik mentés róla. Adott esetben nem kizárt, hogy a felhasználó csak bérli a tárhelyet (outsourcing), ekkor nem kell alkalmazottakat fizetnie.

Több rétegű kliens/szerver architektúra

Az előbb tárgyalt esetben a megjelenítési réteg feladatát megvalósító alkalmazást telepíteni kell arra a gépre, ahol szükség van rá. A vállalattól távol levő telephelyek számára ez azért lehet problémás, mert fejlesztés folyhat a vállalat központjában. Az elkészült új alkalmazás telepítésére el kell utazni a telephelyre. Ha azonban a felhasználói felületet megvalósító alkalmazásokat platform függetlenné lehetne tenni, akkor ez megoldaná ezt a problémát. Ha felhasználói felület biztosítására böngésző programot használnának, akkor jelentősen leegyszerűsödne a helyzet, hiszen valamilyen böngésző program minden gépen, minden elterjedt operációs rendszer esetében áll rendelkezésre. Ekkor még meg kell oldani azt a problémát, hogy a feldolgozás eredményeképp megkapott adatokat olyan formátumra kell hozni, hogy azt egy böngésző program meg tudja jeleníteni. Erre egy Web szerver alkalmas, csakhogy ez egy újabb réteg bevezetésével jár együtt. A felvázolt rendszer kibővíthető több web szerver, több adatbázis szerver, és szükség szerint több alkalmazás szerver bevezetésével. Egy ilyen összetett rendszer igen komoly terhelést képes kiszolgálni. Ha a terhelést nem képes kiszolgálni a jelenlegi rendszer, meg lehet vizsgálni, hol van a szűk keresztmetszet, és megfelelő bővítéssel újra működőképes lesz az egész rendszer. Monolit rendszer, vagy két rétegű kliens/szerver rendszer esetében a teljes rendszert, vagy a központi kiszolgálót kellene lecserélni, lényegesen magasabb költségek mellett.

Adatbázis terv

Az alkalmazás fejlesztésének egyik kulcskérdése.

- A feldolgozandó információ elemzése:
  - ⊗ az adatbázis céljának
  - ⊗ használati módjának meghatározása.
  - ⊗ milyen információkat kívánunk az adatbázistól
- Logikai adatbázis tervezés:
  - ⊗ Egyedek meghatározása
  - ⊗ Egyedeket leíró tulajdonságok megadása
  - ⊗ Egyedek közötti kapcsolatok feltérképezése
  - ⊗ az eredmény ábrázolása (E/K diagram)
  - ⊗ Adatredundancia minimalizálása
- Fizikai adatbázis kialakítás:
  - ⊗ Az adatbázisok létrehozása számítógépen