# 3. Tervezés

## 3.1. Tervezési filozófiák

***Funkcionális (funkcióorientált) :*** folyamatorientált, feldolgozási egységekre koncentráló, output-célú szemlélet, úgynevezett hagyományos tervezési módszer

- Rendszerszervező munka célja : pontosan meghatározni a az előállítandó outputokat
- Adattárolási technikák kezdetlegesek (lyukkártya, lyukszalag)
- Cél : helytakarékosság
- Előtérben a kódolás. A kódolási, tesztelési munkák tették ki a feladatok 65-70 %-át
- Tehát feldolgozó-központos
- Problémák :
  - Redundancia (ez a kifejezés itt került be a köztudatba)
  - Adatok nevei nem egységesen kerültek meghatározásra

***Adatközpontú*:** adatok meghatározásának és kezelésének feladata (cél: redundancia megszüntetése, adatok egységesítése, tehát az előző módszer fejlesztése)

- Nem az output a legfontosabb, mert a felhasználó újabb és újabb outputokat találhat ki
- Megszületett az adatbáziskezelés elve (1974 CODASYL jelentés: adatbáziskezeléssel foglalkozó, nemzetközi bizottság neve)
- Középpontban az adatmodellezés
  - Bachman (adatbáziskezelés első embere)
  - Codd (relációs adatbázis elméletének kidolgozója)
  - Halassy (fogalmi szintű adatmodellezés egységes elméletének megalkotója)

Közben *három fontos momentum* jelent meg, ami az újabb fejlesztési filozófia irányába visz :
- strukturált programozás megjelenése
- projektvezetés módszereinek kidolgozása
- dokumentáció fontosságának felismerése

A fellépő problémák*:*
- hosszú és a tervezetett legtöbbször meghaladó fejlesztési idő
- magas fejlesztési költségek -> kevés a kimutatható eredmény
- elkészült rendszerek nehezen módosíthatók, rugalmatlanok (hiányos fejlesztési dokumentáció, valamint a hibás tervezés okozza)
- felhasználói követelmények gyakran kielégítetlenül maradnak
- új eszközökön gyakran régi rendszer valósul meg

***Struktúrált fejlesztés*** (70-es évek eleje)

- elsőbbsége sem az adatoknak, sem a folyamatoknak, sem az outputoknak nincsen -> minden rendszerelemmel foglalkozni kell, valamint az egymással való kapcsolatukkal is
- lényege: a folyamat eredményeként egy olyan modulokból felépített könnyen áttekinthető és az igényeknek megfelelően javítható terméket kapunk, amely a felhasználó által a fejlesztés elején megfogalmazott igényeket elégíti ki
- a fejlesztés a rendszer adatainak és folyamatainak szerkezetén alapul
- 1975 szoftverkrízis
  - M. Jackson tervezési módszere
  - N. Wirth-Pascal
  - BOEHM szoftverfejlesztési spirálmodellje

- 1977 : SSADM
- 1979 elemzések a módszertanokról : DEMARCO, MYERS

Az ésszerű, szisztematikus módon megvalósított, tiszta, világos szerkezetben, modulokban gondolkodó, a probléma megoldását egyértelmű szakaszokra bontó információ-rendszer fejlesztési tevékenységet és a megvalósítást segítő eszközrendszert strukturált tervezési módszernek nevezzük.

A minőség középpontba kerül. Néhány fontos elvárás :

- Fejlesztési hatékonyság (mennyi idő, milyen erőforrás kell)
- Illeszkedés a problémához (a fejlesztett rendszer kielégíti-e a felhasználói igényeket)
- Kipróbálás, tesztek (voltak-e tesztek, és milyenek lettek az eredmények)
- Megbízhatóság (milyen a hibaelőfordulási valószínűség, megfelelő a hibajavítás?)
- Biztonságos-e (nem vezetnek-e a hibák teljes összeomláshoz)
- Működtetés egyszerűsége
- Dokumentált-e a rendszer
- Működtetési hatékonyság
- Nyitottság, hordozhatóság (biztosított-e az eljárás különböző platformok között)
- Rugalmasság (újonnan felmerült igényekhez lehet-e igazítani?)
- Részek közötti viszony (mennyire integrálhatók a rendszer egyes elemei az egészbe)

Jackson-módszer vezérlőszerkezetei :

1. **Szekvencia**

Sorban végigmegy

$B_1 .... B_n - ig$

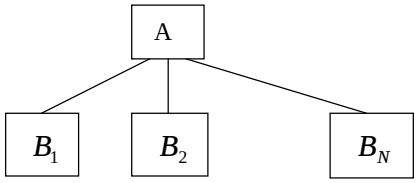
A seq

$B_1$

.

$B_n$

A end




2. **Szelekció**

$A - select - C_1$

$B_1$

$A - or - C_2$

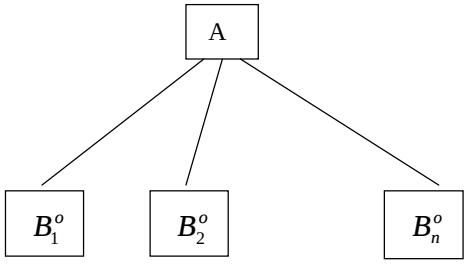
$B_2$

..

$A - or - C_n$

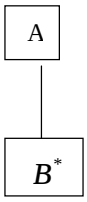
$B_n$

A end



**3. Iteráció**

A iter while C
B
A end




Újabb igények léptek fel :

a) komponens alapú fejlesztés igénye
   - elemekre bonthatóság
   - kombinálhatósági igény
   - érthetőség (egyértelműség a későbbi módosítás miatt)
   - hibák legyenek javíthatók
b) újrafelhasználhatóság biztosítása

***Objektum-orientált tervezés :*** BOOCH, LOKENSEN, WARD, 1997 UML, mint szabvány

- objektum magában hordozza a dinamizmust, amennyiben együtt kezeli az attribútumokat és a műveleteket
- hatékonyan valósítja meg a rendszerkomponensek és folyamatainak modellezését
- alkalmas a valóság absztrakt objektumainak előállítására
- asszociációk komplex leképezésére
- üzleti folyamatok alapos, mindenre kiterjedő elemzésére
- komplex módszertan és eszköz is egyben

- rugalmasság
- időszerűség
- alacsony fejlesztési és megvalósítási költségek
- rövid futási idők, gyors hozzáférés
- alacsony fenntartási és karbantartási költségek
- egyszerű, gyors módosíthatóság
- hordozhatóság és kompatibilitás
- felhasználóbarát használat, szabványos felhasználói felületek
- könnyű, nem szakértők által is lehetséges fejlesztés
- újrafelhasználhatóság
- kibővülő információszerzési lehetőségek

***Tudásbázis-tervezés*** :mesterséges intelligencia rendszereknek az emberi gondolkodás alapján működve képesnek kell lenni arra, hogy a meglevő tudást egy meghatározott cél elérése érdekében át tudják alakítani új ismeretekké. Ez azonban olyan komplex tevékenységet jelent, amelyek során elvégzendők az alábbi feladatok :

- következtetés (létező ismeretből -> új)
- döntéshozatal
- problémamegoldás

Azt a fejlesztési/tervezési módszert, amely többnyire rendkívül komplex, algoritmusokkal le nem írható rendszereket intelligens módon képes megvalósítani, előre nem látható utakat meghatározni, szituációkat, következményeket lejátszani és döntéseket hozni, tudástervezésnek nevezzük.

*Szakértői rendszerek*: azok a rendszerek, amelyek egy nagyon szűk speciális szakterület tudásbázisát tárolják, feldolgozzák és segítségével megoldásokat adnak a problémákra.

Alkalmazások :
- problémamegoldás
- interpretációk
- prognózisok
- folyamatirányítás
- tervezés, ipari feladatok
- orvosi diagnosztika
- geológiai kutatások
- oktatás

## 3.2. Ember-gép kommunikáció

"A felhasználói interfészek tervezése inkább művészet, mint tudomány."

*Felhasználói interfész*: egy mechanizmus, amelyen keresztül a felhasználó kommunikálni tud a számítógépen működtetett alkalmazással. A felhasználói interfész tervezésének célja az, hogy a **felhasználó életminőségét** jobbá tegye az ő számítógépes munkahelyén; továbbá **felhasználóbarát információs társadalom** létrehozása.

Felmerül a kérdés, hogy milyen területeken fontos a felhasználói interfész tervezése:
- szövegszerkesztők,
- szimulátorok,
- CAD (számítógéppel támogatott tervezés),
- www, hypertext.

Középpontban: *az ember* (a felhasználó): figyelembe kell venni a felhasználók pszichológiai, kulturális, nemzetközi különbözőségéből adódó szempontokat:
- ki a felhasználó, milyen a felhasználó:
  - hogyan tudja megtanulni a rendszer használatát
  - milyen a problémamegoldó képessége,
  - milyen a döntéshozásban,
  - keresési gyorsaság a felületen,
- szem előtt kel tartani a tervezéskor a képességeikben korlátozott felhasználók igényeit, melyek a következők:
  - gyengén látók,
  - idősek.
- a nemzetek közötti karakterek, telefonszámok írásának különbözősége,

Felhasználói elvárások a rendszer felé :
- a szükséges időben rendelkezésre áll
- pontos, precíz ellenőrizhető eredményeket készít oly módon, hogy
  - azt a felhasználó érti és követni tudja
  - szükség esetén lehetőség van a beavatkozásra

A felhasználói interfész tervezése során 3 különböző modellt kell kialakítani :

- a munkakörnyezetet és az eszközöket specifikáló feladatmodellt
- a felhasználó feladatokhoz illeszkedő, az interfész elvárt sajátosságait specifikáló felületmodellt
- eszközöket kiválasztó, a kommunikáció, a megjelenítés módját és formáját kialakító kommunikációs modellt

### *3.2.1. Feladatmodell kialakítása*

Feladatmodell = a felhasználói munkahely kialakítása (az ergonómia foglalkozik ezzel)

*Cél :*

- a munkát a felhasználó hatékonyan végezze
- könnyen, hiba és fáradtság nélkül
- a lehető legkisebb energiaráfordítással
- emberközeli, gazdaságos munkakörnyezet

Orvosok, pszichológusok, munkaszociológusok együtt foglalkoznak ezzel, tehát figyelni kell a rendszertervezőknek a következőkre :

- használat közben milyen erőfeszítéseket igényel a rendszer működtetése
- könnyen elérhetők-e az egyes hardver elemek
- látható-e a képernyő különböző fényviszonyok közt
- egyszerűen működtethetők a programok
- ember által érthető természetes kommunikáció zajlik-e az ember és gép között

Napjainkban a hardverfejlesztők is egyre nagyobb figyelmet fordítanak az ergonómiai kérdéseknek :

- színes, villogásmentes képernyők
- elektronikus kisugárzást csökkentő védő-előlapok
- fényceruza
- formatervezett klaviatúra
- grafikus lehetőségek
- egér
- digitalizálók, szkennerek

A munkahelyet is az ember testi és pszichológiai sajátosságait figyelembe véve kell megtervezni. (egészségügyi szempontok; látás védés, hátfájás megelőzése)

Testtartást védése:

- Állítható támlájú szék
- Kézfej tartó
- Állítható asztal

Szem védése:

- Stabil monitor
- Alacsony kisugárzású monitor
- Jó megvilágítás
- Megfelelő távolság beállítása

Fejfájás kiküszöbölése:

- Monitor tisztán tartása
- Fények visszaverődése a monitoron
- Széktámla-monitor távolság helyes beállítása

### *3.2.2. Felületmodell kialakítása*

Szoftverrel szemben támasztott követelmények szoftver-ergonómiai szempontból:

- A működtetés biztonsága
- A manuális és gépi, illetve részben gépi feladatoknak összhangban kell lenniük egymással
- A dialógus akkor éri el célját, ha könnyen érthető, a funkciók egyszerűen vezérelhetők. Mindezt úgy kell megvalósítani, hogy a rendszer kezeléséhez ne terheljük az embert a kommunikációs rendszer sajátosságainak megismerésével.
- Megtanulhatóság : ne kelljen a felhasználónak felesleges dolgokat megtanulnia a szoftverről. Viszont, amire szüksége van, az könnyen elsajátítható kell legyen.

Elvárások a működtetéssel szemben :

- Válaszidő gyors legyen és egyenletes (nem jó, ha hosszát tekintve nagy a változékonyság, mert a felhasználónál nem tud kialakulni jó munkaritmus -> stresszt okoz a hosszú várakozás)
- Felhasználó az aktív alkalmazás nélkül hívhasson segítséget (jó help-súgó rendszer kialakítása)
- Hibák megfelelő kezelése:
  - Érthetően legyenek megfogalmazva
  - Egyértelmű utasítást kell adni a javításra vonatkozóan
  - Fel kell hívni a figyelmet a hibajavítás negatív körülményeire
  - Hibaüzenet kísérje figyelemfelhívó effektus (hang, villogás …)
  - Hibaüzenet ne legyen bántó a felhasználóra nézve

IBM 1987-ben kidolgozott egy szabványgyűjteményt: SAA (System Application Architecture), amely a PC-ket, mini, nagy és szuperszámítógépeket átfogva leszabályozza a különböző gépcsaládok és típusok közötti együttműködés, átjárhatóság feltételeit, módját, technikáját.

SAA legfontosabb fejezetei :

- Egységes grafikus felület : CUA (Common User Acces)
  - Pl.: minden program F1-el hívja a Help-et, milyen formában, hogyan közölje a rendszer a hibaüzeneteket
- Egységes program-interfész : CPI (Common Programming Interface)
  - Növeli a felhasználói programok fejlesztési hatékonyságát (pl.: az egyes adatbáziskezelők által létrehozott állományok más adatbáziskezelőkkel, valamint programnyelvekkel való lekérdezését)
- Egységes kommunikációs támogatás : CCS (Common Communication Support)
  - Arra szolgál, hogy a különböző hardveregységek, programok és hálózatok közötti kapcsolatot az operációs rendszerben egységes módon valósítsák meg.

Megjelenítés (a könnyű kezelhetőség támogatására):

- Menütechnika

- Ablaktechnika (célszerű, ha a rendszer lehetővé teszi, hogy egy időben több munkán is dolgozhasson a felhasználó)
- Ikonok alkalmazása

Az információt a felületen ablakokban jeleníthetjük meg. Az ablak különböző területei más-más funkciót látnak el:

- Az **identifikációs** vagy **azonosítási** területen a megjelenített információ címe található,
- az **I/O területen** az információt fogadja a számítógép a felhasználótól és/vagy megjeleníti azt neki.
- A **kontroll** területen a felhasználó az adott programot tudja irányítani különböző parancsokkal, illetve nyomógombok segítségével.
- Az **üzeneti** területen a rendszer állapotáról talál információt a felhasználó a géptől.

A felület tervezése szempontjából a felületen, illetve az ablakokban fontos részek: a mezők, címkék, jelzők és pointerek vagy mutatók:

- **Mezők:** A mező egy körülhatárolt terület, ahol fix karakterszámú információt vihetünk be beviteli mező esetén, illetve olvashatunk ki kiviteli mező esetén. Fontos, hogy a beviteli mező a kivitelitől vizuálisan különüljön el, valamint bevitelnél a rendszer jelezze nekünk, hogy milyen formában várja az információt, pl. éé-hh-nn.
- **Címkék:** A címke rövid, leíró cím egy objektumhoz (ikonhoz, mezőhöz, stb.) Címkéknél is fontos, hogy különüljön el vizuálisan a jelzett objektumtól.
- **Jelzők:** Az ablakban lehetnek egyszeres vagy többszörös választási lehetőségek, és döntésünk eredményét jelzőkkel jelenítjük meg. Az egyszeres választás jelzője különbözzön a többszörösétől.
- **Mutatók:** Ilyen, pl. az egér mutatója. A háttértől különüljön el a mutató.

Az ablakban megjelenített információ jellemzői:

- tömörség (mindig csak az adott feladathoz szükséges információ jelenjen meg),
- konzisztencia (mindig ugyanolyan formában jelenjen meg az információ),
- értelmesség (az adott feladathoz kapcsolódó információ jelenjen meg).

Az információt szervezhetjük listákba vagy táblázatokba is. Ezekre vonatkozó követelmények:

- az információ sorrendezése logikai vagy természetes legyen, ha ezek közül egyik sem valósítható meg, akkor abc sorrendezést válasszunk,
- a nem numerikus információt balra, míg a numerikus információt jobbra igazítsuk,
- táblázatokban a bal oldali oszlopban legyenek a legfontosabb információk.

Színek használata a felületen:

- ne használjunk túl sok színt, mert attól zavaros lesz a képernyő és így a felhasználó keresési ideje hosszabb lesz,
- maximum hat színt használjunk kategorizálásra, csoportosításra,
- konzisztensen használjuk a színeket.

A kommunikációs modell tervezésénél (fejlesztésénél) az alábbiakat kell figyelembe venni :

- Következetesség : azonos formát alkalmazzunk a menü kiválasztásánál, parancsok bevitelénél, adatmegjelenítésnél

- Visszacsatolás : egyes feladatoknál kérjünk visszacsatolást a felhasználótól, hogy megértette-e a dolgokat
- Biztosíték : vissza nem állítható műveleteknél kérjünk megerősítést (Valóban törölni akar?)
- Minimalizáljuk : azon paraméterek és teendők számát, amit a felhasználónak két művelet végrehajtása közben meg kell jegyezni
- Bocsássuk meg a vétett hibát : a rendszer ne hajtsa végre a hibát okozó utasítást
- Vezérléshez alkalmazzunk rövid, egyszerű, könnyen megjegyezhető szavakat

***Inputok tervezése***

*Bizonylatok:*
- Felülről-lefelé, balról-jobbra kelljen kitölteni
- Nem szabad összekeverni a bizonylati információkat a kitöltendő adatokkal
- Szét kell választani azokat az adatokat, amik majd gépi rögzítésre kerülnek
- Minimalizálni kell a bizonylatra írandó adatokat, ne kérjünk be olyan adatot, ami a gépben már tárolva van, vagy pedig az addig rendelkezésre álló adatokból kiszámolható

A legtöbb bizonylat a gépi adatfeldolgozás előtt egy előfeldolgozáson megy végig :
1. Ellenőrzi a bizonylatokat helyesen töltöttük-e ki (pl.: minden mező ki van-e töltve)
2. Pontosan elő kell írni az adatátvitel módját, idejét, felelősöket
3. Rögzítésről, a munka során adódó hibákról, a körülményekről pontos nyilvántartást kell végezni

*On-line adatbevitel:*
- Minimalizálni kell a felhasználó által bevitt adatok körét
- Bevitt adatokat meg kell jeleníteni, ellenőrizni, és a hibás adatot és a vélt hibát is ki kell írni
- Az adatbeviteli és az eredményeket közlő képernyő megjelenítésénél is legyünk konzisztensek : azonos elrendezést, színeket, formát használjunk
- Ne a számítógépben használt belső kódokat használjuk (ne terheljük feleslegesen a felhasználót), hanem azok természetes formáját szövegeket jelenítsünk meg
- Inaktiválni kell azokat a mezőket, amelynek végrehajtása az adott kontextusban értelmetlen
- Külön jelöljük meg a kötelezően kitöltendő mezőket (gyakorlat erre a *)

***Dialógusok tervezése :*** felhasználó-számítógép párbeszéd

A felhasználó a számítógéppel dialógusokon keresztül kommunikál. A dialógustól nagymértékben függ az ember-gép kölcsönhatás sikeressége.

**Norman-féle ember-számítógép kölcsönhatás modell:** Egy interaktív rendszer célja, hogy segítsen a felhasználónak céljai megvalósításában az alkalmazási területeken. A modell elemei: domain (szakmai terület, maga a szakma), task (az elvégzendő feladat, művelet), goal (cél, ami a kívánt megvalósítása a feladatnak).

**Végrehajtási-kiértékelési ciklus:** A felhasználó a tervét a számítógépen végrehajtja, majd megvizsgálja az eredményt, hogy egyezik-e a céllal. Ennek a lépései:
- célok meghatározása a szakma nyelvén,

- végcél kialakítása (pontosítjuk a célt),
- műveletek sorrendjének meghatározása,
- műveletek végrehajtása,
- rendszer állapotának észlelése,
- rendszer állapotának lefordítása/értelmezése,
- a célokra és a végcélokra vonatkozóan kiértékeljük a rendszer állapotát:

**Végrehajtási szakadék a Norman-modellben:**
Különbség azon műveletek között, amiket a felhasználó megtenne céljai elérése érdekében és a rendszer által engedett (ténylegesen megvalósítható) közt. Ha a különbség nem nagy, akkor a kölcsönhatás sikeres lesz, ellenkező esetben nem. **A felhasználói felület tervezésének célja, hogy a végrehajtási szakadékot csökkentse!**

**Kiértékelési szakadék a Norman-modellben:**
A felhasználó elvárásai és a gép által megjelenített eredmény közti különbséget nevezzük kiértékelési szakadéknak. Ha a megjelenített eredményt könnyen ki tudja értékelni a felhasználó, akkor a kiértékelési szakadék kicsi. Minél nagyobb erőfeszítésébe kerül a felhasználónak az eredmény lefordítása, annál nagyobb a szakadék.

Dialógustervezésnél az alábbiakat kell figyelembe venni:
- Felhasználónak mindig tudnia kell, mikor és merre léphet tovább, hogyan kapja meg az előző dialógusokat, azaz a végrehajtási folyamat éppen melyik fázisában van
- Képernyő mindig ugyanolyan struktúrájú legyen (színű stb…)
- Egy képernyőablak egy megfelelő feladatra legyen alkalmas
- Üzeneteket, információkat, utasításokat tartalmazó ablakokat olyan méretűre kell választani, amit a felhasználó kényelmesen olvashat
- Óvatosan kell bánni a fényeffektusokkal (csak a fontos információknál, figyelemfelkeltésnél kell használni)
- Alkalmazzunk alapértelmezéseket, automatikusan vezérelt eljárásokat -> természetesen előre egyeztetve a felhasználóval
- Funkciók aktivizálásakor ne alkalmazzunk nehezen megjegyezhető, eltéveszthető billentyűzet-kombinációkat
- Nyelvtanilag helyes, érthető kifejezéseket használjunk
- Óvakodjunk a vicces megjegyzésektől

**Főbb dialógustechnikák:**
- **Parancssoros felület:** A dialógus során a felhasználó parancsokat ad ki a számítógépnek.
    - Előny: gyakorlott felhasználóknak gyors hozzáférést biztosít a rendszerhez, valamint rövidítéseket lehet használni.
    - Hátrány: kezdő felhasználóknak nehéz megtanulni, memorizálni a parancsokat.

- **Menü** (A dialógus rendszer felkínál a felhasználónak egy vagy több parancscsoportot, a felhasználó választ egy vagy több parancsot, és a számítógép végrehajtja a parancs által jelölt folyamatot) **felület:** A felhasználó által elérhető parancsok (halmazának) megjelenítése a képernyőn -> kiválasztásuk egérrel, vagy billentyűzettel.
    - A menü elérésének módjai:
    - parancsgombokkal vagy ezek rövidítésével,
    - billentyűgomb megnyomásával,

- egér-klikkel,
- hanggal.
  - Az opció kiválasztásának módjai:
- gyorsbillentyűkkel,
- az opció számának vagy kódjának begépelésével,
- a kiválasztott opció megérintése a képernyőn.
  - Menün belül az opciók sorrendezése:
- konzisztens,
- fontossági,
- megállapodás szerinti,
- használati gyakoriság szerinti,
- abc.

    - Előny: parancsok láthatók (kevesebbet kíván a felhasználótól tanulás szempontjából), valamint az, hogy a felismerésre épít, nem az emlékezésre.
    - Hátrány: a menük gyakran hierarchikus felépítésűek, így a felhasználó lassabban találja meg azt, amit keres.

- **Űrlap kitöltési és nagy táblázatos felület:** Elsősorban adatbevitelre használják, de adatok visszanyerésére is hasznos lehet. A felhasználó egy papírlap formátumú képernyőt lát, az üres helyeket kell kitöltenie -> van javítási lehetősége.
    - Előny: könnyű megtanulni és használni kezdőknek.
    - A nagy táblázatok speciális formái az űrlapkitöltési dialógusoknak (pl. Excel). Formulákat vihet be a felhasználó a cellákba => rugalmas felület.

A felületet használhatóság szempontjából is vizsgáljuk. Ennek három fő kategóriája:
- tanulhatóság (az új felhasználó hamar meg tudja tanulni a felület használatát => a felhasználó hatékonyabb lesz),
- rugalmasság (a módok sokféleségét jelenti),
- robosztusság (a felhasználó támogatottságának mértékét jelenti -> help-ek).

A felhasználói felület használhatóságának vizsgálatára a következő módszerek ismeretesek:
- heurisztikus vizsgálat (megítéli, hogy minden dialóguselem megfelel-e a felállított használhatósági elveknek),
- konzisztencia vizsgálat (következetesség szempontjából vizsgálja a felületet),
- szabvány vizsgálat (a felület megfelel-e a szabványnak).

***Output tervezés***

Képernyőtervezés :
- Csak azok az információk szerepeljenek, amelyek az adott kérdéshez tényleg hozzátartoznak
- Az adatokat úgy jelenítsük meg, hogy azonnal érthető legyen (ömlesztett adatok helyett -> grafikon)
- Tegyük lehetővé az eredmények vizuális megjelenítését (ábrák, rajzok, képek)
- Az elrendezésben is igazodjunk a megjelenítendő lényeghez

Papíralapú listák :

- Lista adattartalmának meghatározása : össze kell gyűjteni a felhasználó számára fontos adatokat, definiálni kell, hogy szöveges információk kellenek-e vagy kódok (ilyenkor minden oldalon definiálni kell a kódokat)
- Lista rendezettsége : az adatok megjelenítési sorrendjének meghatározása (növekvő-csökkenő)
- Definiálni kell olyan fontos jellemzőket, mint : készítés időpontja, felelőse stb…
- Lista formai tervezése
  - Fejlécek kialakítása (lista tartalmára és céljára utaló cím, készítőjének neve, oldalcím, dátum)
  - Lapváltás feltételeinek definiálása (elő kell írni azokat a feltételeket, amelyeknél mindenképpen új lapot kell kezdeni)