Rendszertervezés

A legtöbb hallgató már hallgató korában lehetőség kap arra - főleg itt, az Adatbázisok Labor keretein belül -, hogy rendszereket hozzon létre a tervezéstől és a megvalósításig. Igazából ez az egyetlen olyan munka, amely egy mérnököt mérnökké tesz. Sajnálatos módon azonban, a rendszertervezés kimaradt az oktatási alapanyagból. Tíz percben nem tudnám összefoglalni a lényegét, de legalább megpróbálok egy vázat mutatni arról, hogy miből is kell álljon egy rendszerterv, és ami talán még fontosabb: miért.

A rendszertervezés célja, hogy egy jól működő rendszert álmodjon meg. A legtöbb hallgató (sőt, végzett mérnökből is túl sok) abba a hibába esik, hogy egy problémát az első kínálkozó módon igyekszik megoldani, nem törődve azzal, hogy milyen hosszú távú következményei lesznek majd a döntéseinek. Ezt a fajta gondolkodást hagyjuk meg a politikusoknak, jogászoknak és közgazdászoknak. Nem az a legfontosabb, hogy egy probléma meg legyen oldva, hanem az, hogy ideális megoldást találjunk rá. Két mérnököt egymástól, illetve a mérnököt a technikustól éppen az különbözteti meg, hogy mennyire jó az a megoldás, amelyet egy problémára tudnak adni, azaz mennyivel több szempont szerint tud kielégítő megoldást vagy lehetőségeket kínálni **a közvetlen problémamegoldáson túl**.

Mérnöknek lenni annyit jelent: előre látni, hogy egy megvalósítandó rendszer/termék hogyan működik, viselkedik majd a hétköznapokban.

Egy tipikus példa a rendszertervezés hiányára az - ezzel szinte minden szoftverfejlesztő találkozott -, amikor egy szoftveralkalmazás átadása után a megrendelő szembesül azzal a problémával, hogy fogalma sincs, mit és hogyan csinál a szoftverünk, mit, hol, hogyan lehet benne elérni, használni. Úgy érzi, a rendszer nem csak nehezen használható, de még körülményesebb is, mint a korábbi, jól megszokott módszer szerint dolgozni.

Az alábbihoz hasonló helyzettel is sok hallgató találkozott már bizonyára. Amikor elkészült végre a nagy alkotás (pl. házi feladat), de még valami kis nüansznyi dolgot kell módosítani a programkódon (pl. beszúrni egy összeadást egy váratlan szempont szerint), akkor az percek helyett csak napok alatt sikerül, mert az egész programstruktúrát lényegében át kell írni a módosításhoz.

Az ilyen és a számtalan hasonló helyzet elkerülésének kulcsa az, hogy körültekintően járjunk el már a program kialakításának a legelején is. Számítsunk változásokra, problémákra, hibákra - ezek mindig lesznek -, és készüljünk fel megfelelő módon azok kezelésére. A legtöbb hallgató, mérnök, informatikus ilyenkor persze azt vágja a fejemhez, hogy szorít az idő, meg gyorsan kell mindig mindent csinálni; kinek van ideje alaposan tervezni. A tervezés az informatikai rendszerfejlesztés elején valóban többlet ráfordítást jelent, azonban ez bőven kompenzálódik nem csak a fejlesztési (például a párhuzamosíthatóság kapcsán), a dokumentációs (felhasználói és oktatói kézikönyvek alapja maga az absztrakt terv) és a tesztelési szakaszban (látjuk, mit, mikor, hogyan kell ellenőrizni), de annak utóélete során is (termékfejlesztés, marketing, testre szabás). A harmadik-negyedik alkalommal pedig olyan rutinná válik az összetett tervezés, hogy a többlet-ráfordítást alig fogjuk megérezni.

De miből is áll a rendszerterv? - legalábbis szerintünk.

- **Funkcionális terv**. A funkcionális tervnek a lényege az, hogy koncepcionálisan, nagy vonalakban, ha úgy tetszik 0. szinten végiggondoljuk (lásd Programozás technológiája), milyen szereplők (aktorok), funkciók (szolgáltatások, feladatok), erőforrások (adatbázisok, eszközök, szoftverek) és komponensek kerülnek egymással kapcsolatba. A funkcionális terv az egyes elemekről leírja
    - A teljes rendszer célját, motivációját, helyét a világban (Bár ez a legfontosabb rész a szereplők igényei mellett, mégis sok dokumentumból kimarad. Fontos azonban látni, hogy a célok, a valós életbeli felhasználása az, ami alapján két lehetőség közül választunk, vagy valami mellett döntünk. Nagy, intézményi projekt esetében itt kell szerepeljen, hogy ez hogyan illeszkedik az intézmény stratégiájába. Szokás ezt vezetői összefoglalásnak, kitekintőnek is nevezni. Terjedelme max. 3-5 oldal.)
    - Az egyes szereplők igényeit, céljait, szempontjait, ahogyan a rendszert használni szeretné (Szokás ezt igényspecifikációnak is nevezni. Terjedelme rendszerenként nagyon változó, a szereplők, szereplőtípusok számától jelentősen függ. Átlagosan, tapasztalataim alapján nagyjából 2-3 oldal/szereplő.)
    - Az egyes komponensek szerepét, felelősségi és hatáskörét (Ügyelni kell arra, hogy a komponensek minél kisebb méretűre tervezzük, de mégis teljes feladatokat oldjanak meg. Ez egyrészt azért fontos, mert célfókuszált, de se nem túl speciális, se nem túl általános komponenseket nagy valószínűséggel később újra fel tudjuk majd használni; másrészt pedig a feladatok párhuzamosítása, szétosztása, illetve a függetlensége a rendszer későbbi fejlesztése, javítása és ellenőrzése szempontjából ez az egyetlen kulcsfontosságú tényező. Komponensenként 0,5-1 oldalban összefoglalható a lényeg.)
    - A komponensek minimális információigényeit, be- és kimeneti mezőit és állapotváltozóit (Lásd Jelek és rendszerek c. tárgy négypólusú világról szóló nézetét. A változóktól való függés, információs igény általában a komponensek felelősségéből, szerepköréből általában származtatható, ezért azokkal összevontan kezeljük. Lényegében a legfontosabb interfész elemeket már itt definiáljuk, de nem megyünk abba bele, azok pontosan hogyan is nézzenek ki. Ez ugyanis függ még a teszteléstől és az adatbázisok szerkezetétől is -- esetleg a kapcsolódó szabványoktól, készként átvett szoftverkomponensektől.)
    - Az egyes komponensek által használt erőforrások listáját, illetve azok specifikációját (Ha van korábbról hozott, ismert komponens, akkor azok felületének leírását, szabványok megnevezését és azok referenciáit, valamint a felhasználható adatbázisok, adattárak szerkezetét.)
    - A megvalósítás tervezett programnyelvi környezetét és operációs rendszerét - részletes indoklással (Számos projekt a "szeretett" programozási nyelven és operációs rendszerre íródik még akkor is, ha van értelme más lehetőségen is eltöprengeni. Sok cég egyetlen programnyelvre specializálta magát, nekik nyilván nincs más alternatívájuk, bár a termékskálát sosem árt bővíteni. Minden más esetben azonban szükség van arra, hogy a megrendelő fejével gondolkodjunk és azt lássuk, hogy neki mire van szüksége; milyen költséggel, infrastruktúrával, saját erőforrással tudja-e üzemeltetni. A döntés hátterét indokolni, okait a munka folytatásához, környezetének megismeréséhez,

elkészültének körülményeinek megértéséhez felettébb szükséges. Terjedelme 1-3 oldal.)

- **Tesztelési terv**. A tesztelési terv a szoftver üzembiztonságának, az ellenőrzésének, egyszersmind az átadásának követelményeit rögzítik. A tesztelési tervet a hallgatók és a rosszabb szoftverfejlesztő cégek szinte mindegyike kihagyja. Az ok, persze, egyszerű: sok, aprólékos munka kell az elkészítéséhez. Miért hiba? Azért, mert egy jó teszterv segíti a fejlesztőt a gyors haladásban (tudja, mire kell figyelnie), láthatóak a kivételkezelés legfontosabb módozatai, tudjuk, hogy milyen körülmények és paraméterek között szabad futtatni a szoftverrendszert, illetve az átadás-átvétel is könnyebben fogadtatható el a megrendelővel, hiszen rögzített, mit kell tudnia kezelni a rendszernek. A tesztelési tervnek egy jórészét nem a fejlesztőknek, hanem a megrendelőnek, vagy a megrendelővel közösöen kell/érdemes kialakítani. Mindegyik terv mérete erősen függ a rendszer bonyolultságától - irányelvként 5-10 oldal/alacsony szintű komponenst érdemes rá számolni. A tesztelési terv elkészítéséhez legalább négy különböző nézőpontból érdemes gondolkodni.
    - Gondolkodjunk kifejezetten a funkcionalitás teszteléséről és tesztelhetőségéről. Minden egyes modul számára, esetleg a rendszer egésze számára tipikus helyes és tipikus helytelen(!) tesztvektorokat, bemeneti tesztadatokat érdemes gyártani. Ha nem túl költséges, akkor kimerítő tesztelést is végezhetünk. Rögzített véletlenszerű vagy a legfontosabb funkciók tesztelésére egyedi és kombinált megoldásokat is készíthetünk. Utóbbi azért lehet jó, mert nem csak azt detektálja, hogy hol a hiba, hanem azt is, hogy milyen funkció feldolgozása helytelen. A funkcionális terv legmeghatározóbb elemeire is érdemes tételes, objektívan ellenőrizhető tesztfeltételt, tesztadatot készíteni. Minden bemeneti tesztadat mellé az elvárt kimeneti adatot (vagy jelenséget) is rögzíteni kell.
    - Készítenünk kell egy hibatűrési tesztet is, ami főleg magyar nyelvterületen lehet kritikus. Tesztelni kell azt, hogy hogyan viselkedik a rendszer nem rendeltetésszerű használatra - nem okoz-e károkat esetleg egy képzetlen felhasználó, vagy egy hirtelen fellépő nagy terhelés. Az utóbbit szoktuk stressz tesztnek nevezni. A hibatűrés ellenőrzésének része szokott lenni az ellenségteszt, amikor kifejezetten a rosszindulatú támadások célpontját meghatározzuk. A teszt célja annak felderítése, hogy a kész rendszernek hol vannak gyenge pontjai. Nem baj, ha van ilyen, de az baj, ha ennek nincs a fejlesztő tudatában, azaz nem ismeri őket.
    - A kész rendszer kiszolgálási ideje lehet kritikus. Például egy felhasználókat folyamatosan kiszolgáló szoftverrendszer válaszideje nem lehet több 1-2 másodpercnél, mert a felhasználó kényelmetlennek fogja érezni a termékhasználatot. Éppen ezért a hatékonyság mérésére is érdemes objektív méréseket, teszteseteket előállítani. A teszt célja az, hogy egyrészt a kritikus részeket azonosítsuk, másrészt a javítást és a javulást kimutathassuk. Sok esetben a konkurens termék hasonló tesztjét is célszerű elvégezni.
    - Végül, de nem utolsó sorban a szoftver elfogadtatásának, valós környezeti tesztelését is érdemes megtervezni. A teszt célja, hogy a felhasználás zökkenőmentességét, elégedettségét, és általános, nem csak a funkcionális használhatóságát mérje. A teszterv általában azt tartalmazza, hogy kik, milyen körben, hol fogják a belső, ún. alfatesztelést, illetve a félnyilvános bétatesztet végezni. A terv hiányában projektfinanszírozási problémák is felmerülhetnek(!).

- **Üzemeltetési terv**. Az üzemeltetési terv készítésének kiindulási pontja az a helyzet, amikor a rendszer -- előzetesen, fejben és papíron -- működik. A terv elkészítése során olyan kérdésekre kell választadnunk, hogy a rendszert körülvevő személyzet hogyan fogja működtetni a rendszert, illetve tanítani a rendszerhasználatra a felhasználókat. A terv nagyrésze nem kritikus, azaz nem feltétlenül része teljes egészében a rendszertervnek. Két területre érdemes tehát fókuszálni az üzemeltetési tervet:
    - A rendszert felügyelő felhasználók hogyan fogják monitorozni, karbantartani, illetve vezérelni vagy szabályozni a rendszert. A terv egyfelől a naplózási és megjelenítési funkciókat, másrészt a rendszer szabadon állítható paramétereinek körét, illetve az állító felület kialakításának sajátosságait foglalja magában. Az üzemeltetési terv ez a fele mindenképpen benne kell legyen a rendszertervben, hiszen hibajelzéseket, kivezetéseket, szabad paramétereket már előre meg kell határozni. A terv programozói dokumentáció és az adminisztrátori kézikönyv legfontosabb elemeit is szokta már tartalmazni. (3-5 oldal/alacsony szintű komponens).
    - A rendszer felhasználását segítő tréningek, tanfolyamok oktatási anyagainak legfontosabb elemeit is érdemes összeállítani. A terv azért lehet fontos, mert ha az átadás környéként derül ki, hogy a szoftver használatához nincs megfelelően képzett munkatársa a cégnek, akkor sem a tesztelés, sem a termék megfelelő felhasználása nem garantálható. Ilyen esetben szinte bizonyosan újra kell írni a szoftvert, jelentősen leegyszerűsítve a működését. Az oktatási tananyagok előkészítése abban is segít, hogy a tervezési fázis során jobban megértsük a célkörnyezet működését, a felhasználók gondolkodását.
- **Adatszerkezeti terv**. A legtöbb alkalmazás hátterében részben a csatolófelületek, részben a működési folyamattól független, önálló tevékenységet nem végző passzív eszközök, a tárolók állnak. Ezek kapcsán beszélünk interfész és adatbázis tervről. Mindkettő csak azután rögzíthető, miután a fenti tervek mindegyikét rögzítettük - az esetek többségében a korábban elkészítendő tervekből levezethető.
    - Az interfész terv jelentősége az, hogy a rendszer egészét, párhuzamos fejlesztését, illetve a korábban már említett funkciók beépítését lehetővé tevő interfészeket meghatározzuk és rögzítsük. A terv mindig a megvalósítás minimális szeletét tartalmazza. Pl. objektumorientált programozás esetében az interfész terv az osztályok minimálisan szükséges definícióját rögzíti.
    - Az adatbázis terv a passzív erőforrások egységes leírását foglalja magába. A nevével ellentétben nem csak az adatbázis, hanem a naplófájlok definícióját is itt szokás megadni. Természetesen, a naplófájl tartalma erősen függ az üzemeltetés sajátosságaitól. Mivel a passzív erőforrások tipikusan közösen és függetlenül használt erőforrások, így a rendszertervből nem maradhatnak ki, enélkül a komponensek fejlesztése nem indulhat meg. Nincs rendszerterv adatszerkezeti terv nélkül!
- **Megvalósítási terv**. A megvalósítási terv az oktatásban leginkább bemutatott, illetve a legtöbb irodalommal rendelkező terület. Számos módszertan, eszköz, támogatás van a jó megvalósítási terv elkészítéséhez. Éppen ezért a legfontosabb elemeit csak címszavakban említjük meg. A megvalósítási terv tartalmaz:
    - magas szintű funkcionális folyamat ábrát, amely az idő függvényében mutatja be a program jellemző folyamatait pl. funkcionális dekompozíció segítségével;
    - időzítési vagy adattovábbítási diagramot, amely az egyes komponensek közötti üzenetváltásokat jellemzi;
    - a bonyulaltabb, nem triviális funkciók, eljárások pszeudokódját;

    - a megvalósítandó osztályok teljes definícióját, amely szuperhalmaza az adatszerkezetnél rögzített interfészeknek;
    - végül az ún. use-case diagramok, amelyek egy adott példán keresztül illusztrálják a helyes (vagy helytelen) működést.
- **Képernyő terv**. Iparművészeti hajlamaink kiélését szolgáló dokumentum, sok kép, alatta bőséges, nem szószátyár magyarázat. A terv legfontosabb tulajdonsága, hogy rögzíti, mit, mikor, hol és hova kell helyezni és egyetlen képernyőnyi méretbe belezsúfolni, illetve mely ablakokból, felületekből milyen felületek, ablakok érhetőek el - elsősorban a use-case diagramokhoz, valamint a célfelhasználók szokásaihoz igazítva. A terv célja elsősorban ezen paraméterek meghatározása, a többit szakavatott, ergonómiához és dizájnkérdésekhez jobban értő szakemberre illik bízni.