2 SZOCIOTECHNIKAI RENDSZEREK

TÉMAKÖRÖK

A fejezet feladata az, hogy bemutassa a szociotechnikai (embert, hardvert és szoftvert tartalmazó) rendszerek tervezésének alapkoncepcióit, és kifejtse, megvitassa annak rendszertervezési folyamatát. A fejezet elolvasása után az alábbi dolgokkal leszünk tisztában:

- megtudjuk, hogy mi a szociotechnikai rendszer, és megértjük, mi a különbség a szociotechnikai és a technikai számítógép-alapú rendszerek között;
- megismerjük az eredendő rendszertulajdonságok fogalmait; ilyenek például a megbízhatóság, a teljesítmény, a biztonság és a védelem;
- megértjük a rendszertervezési folyamat tevékenységeit;
- megértjük, miért van hatással a szervezeti környezet a rendszer tervezésére és használatára;
- megtudjuk, mit jelent az „ősrendszer", és azt, miért kritikus az ilyen rendszerek használata.

TARTALOM

A *rendszer* szó az egyike a legszélesebb körben alkalmazott szavunknak. Beszélünk számítógéprendszerről, operációs rendszerről, fizetési rendszerről, oktatási rendszerről, közigazgatási rendszerről stb. Ezek mindegyike nyilvánvalóan egy eltérő használatát takarja a *rendszer* szónak, de mindegyik magán viseli a *rendszer* szó azon tulajdonságát, hogy az egy olyan valami, ami több, mint az őt alkotó részek puszta együttese.

A nagyon absztrakt rendszerek, mint például a közigazgatási rendszerek, eléggé kívül esnek ennek a könyvnek a témakörén. Következésképpen én csak olyan rendszerekre fokuszálok, amelyek számítógépeket tartalmaznak, és valamilyen speciális célt szolgálnak, például biztosítják a kommunikációt, támogatják a navigációt vagy épp kiszámolják a béreket. Az ilyen rendszerek definíciójának egy számunkra hasznos munkaverziója így a következő lehet:

A rendszer egymással kölcsönösen kapcsolatban lévő komponensek jól átgondolt, egy adott cél elérése érdekében együtt dolgozó együttese.

Ez az egyszerű definíció rengeteg rendszert foglalhat magában. Például egy nagyon egyszerű rendszer a toll, mely három vagy esetleg négy hardverelemből áll. Ezzel ellentétben például egy forgalomirányítási és ellenőrzési rendszer több ezer hardver- és szoftverkomponenst rejt magában, és emberi felhasználókat, akik döntéseket hoznak a számítógépes rendszer által szolgáltatott információk alapján.

A szoftvereket tartalmazó rendszerek két kategóriába sorolhatók:

- *A technikai számítógép-alapú rendszerek* olyan rendszerek, amik hardver- és szoftverkomponensekből állnak, de nem eljárásokból és folyamatokból. Az ilyen technikai rendszerekre jó példa a televízió, a mobiltelefon és természetesen a legtöbb személyi számítógépes szoftver. Az egyének és a szervezetek különféle célokból alkalmazhatnak technikai rendszereket, de az adott célhoz tartozó tudáshalmaz nem képezi a rendszer részét. Például a szövegszerkesztőnek, amit éppen használok, nem része a könyv, amit éppen a segítségével írok.
- *A szociotechnikai rendszerek* olyan rendszerek, melyek tartalmaznak egy vagy több technikai rendszert, de létfontosságú dolog, hogy azon túl tartalmaznak egy tudáshalmazt is arról, hogyan szolgálják ők egy tágabb cél elérését. Ez azt jelenti, hogy ezek a rendszerek jól definiált működési folyamattal rendelkeznek, beleértve a (működtető) embereket, mint a rendszer részeit, illetve a szervezeti szokásokat, szabályokat, külső megszorításokat (például törvényeket), melyek a működést befolyásolják. Például ez a könyv egy szociotechnikai kiadói rendszeren keresztül születik meg, mely különféle folyamatokkal és technikai rendszerekkel rendelkezik.

A szociotechnikai rendszerek fontos jellegzetességei az alábbiak.

1. Olyan eredendő tulajdonságokkal rendelkeznek, melyek inkább a *teljes* rendszerre vonatkoznak, mintsem annak különálló részeire. Az eredendő tulajdonságok nagyban függnek mind a rendszerkomponensektől, mind a köztük lévő kapcsolatoktól. Mivel ezek eléggé összetettek, így az eredendő tulajdonságok csak akkor értékelhetők ki, ha az egész rendszer összeállt.
2. Gyakran nemdeterminisztikusak. Ez azt jelenti, hogy nem biztos, hogy egy adott bemenetre ugyanazt a kimenetet fogják szolgáltatni. A rendszer viselkedése függ az emberi operátortól is, és az emberek nem mindig ugyanúgy reagálnak. Sőt a rendszerek használata új kapcsolatokat is teremthet a rendszerkomponensek között, így a változás maga is egy eredendő viselkedés.
3. Az, ahogy a rendszer kiszolgálja a szervezeti célokat, nem csak a rendszertől magától függ. Ez nagyban függ a célok stabilitásától, a szervezeti célok közötti kapcsolatoktól és konfliktusoktól, és attól is, hogyan értelmezik ezeket a célokat az emberek. Az új menedzsment újraértelmezheti a szervezeti célokat, és így a korábbi célokat támogató „sikeres" rendszer is lehet „hibás".

Ebben a könyvben az olyan szociotechnikai rendszerekkel foglalkozom, amelyek hardver- és szoftverelemeket tartalmaznak, illetve a szoftver által implementált interfésszel is rendelkeznek a felhasználók felé. A szoftvertervezőknek, az ilyen rendszerekben található szoftverek fontossága miatt, számos ismerettel kell rendelkezniük a szociotechnikai rendszerek rendszertervezésének területéről (White *és mások*, 1993; Thayer, 2002). Például kevesebb mint 10 MB szoftverkódra volt szüksége a US Apollo-programnak ahhoz, hogy 1969-ben embert juttasson a Holdra, de körülbelül 100 MB szoftverkódra volt szükség a Columbus űrállomás vezérlőrendszeréhez.

A rendszerek jellegzetessége nem más, mint a rendszerkomponensek tulajdonságainak és viselkedésének kibogozhatatlan keveréke. Minden rendszerkomponens működése függ más komponensek működésétől. Tehát a szoftver csak akkor működhet, ha a processzor is működőképes. A processzor pedig csak akkor hajthat végre számításokat, ha a sikeresen telepített szoftverrendszer definiálja, mik ezek a végrehajtandó számítások.

A rendszerek túlnyomó többsége hierarchikus felépítésű olyan értelemben, hogy más rendszereket is magában foglal. Például egy rendőrségi parancsnoki ellenőrzési rendszer tartalmazhat akár egy földrajzi információs rendszert is abból a célból, hogy részleteket szolgáltathasson egy incidens helyét illetően. Ezeket az egyéb rendszereket *alrendszerek*nek hívjuk. Az alrendszerek független létjogosultságú rendszerekként működnek, ez adja meg jellegzetességüket. Ily módon ugyanaz a földrajzi információs rendszer más rendszerekben is használható.

Mivel a szoftver természeténél fogva flexibilis, számos nem várt probléma leselkedik még a szoftverfejlesztőkre. Tegyük fel, hogy a radar bizonyos helyzetben szellemképeket produkál. Mivel a radar mozgatása nem praktikus, így más mód szükséges a szellemképek kiküszöbölésére. A megoldás az lehet, hogy fokozzuk a szoftver képfeldolgozó képességeit a szellemképek eltüntetésére. Ez később nagyobb processzorteljesítményt kíván, amelynek biztosítása nehézségekbe ütközhet.

A szoftvertervezők gyakran a szoftverképességek növelésével kívánják megoldani a problémákat anélkül, hogy megnövelnék a hardverköltségeket. Számos úgynevezett „szoftverhiba" nem a hozzá tartozó szoftver problémáinak következménye, hanem a módosított rendszerkövetelményekhez való illeszkedés érdekében történő szoftverváltoztatások eredménye. Jó példa az ilyen típusú meghibásodásokra a denveri repülőtér poggyászkezelő rendszere (Swartz, 1996).

A szoftvertervezés tehát kritikus az összetett, számítógép-alapú szociotechnikai rendszerek sikeres fejlesztése szemszögéből. Szoftvertervezőként nemcsak a szoftverekkel magukkal kell foglalkoznunk, hanem tágabban kell szemlélnünk a problémát, és azt is figyelembe venni, hogyan fog a szoftver kapcsolatba lépni más szoftver- és hardverrendszerekkel, és hogy feltehetően hogyan fogják azt használni. Ez a tudás lesz majd segítségünkre abban, hogy megértsük a szoftverek határait, hogy jobb szoftvereket tervezzünk, és hogy egyenlő partnerei legyünk egy szoftvertervezői csapat tagjainak.

2.1. AZ EREDENDŐ RENDSZERTULAJDONSÁGOK

A rendszer komponensei közötti komplex kapcsolatok arra utalnak, hogy a rendszer több mint a részek puszta együttese. Egy rendszer olyan tulajdonságokkal is rendelkezik, amely csak a rendszer egészére jellemző. Ezeket az *eredendő tulajdonságok*at (Checkland, 1981) nem köthetjük egyik különálló rendszerrészhez sem. Ezek az eredendő tulajdonságok elsősorban akkor jelennek meg, amikor a rendszer komponenseit integráljuk. Néhány ilyen tulajdonság persze származtatható valamelyik alrendszer tulajdonságából, de sokkal gyakoribb, hogy megjelenésük az egyedi komponensekkel nem, csak a köztük lévő komplex kapcsolatokkal magyarázható. Az eredendő tulajdonságokra a 2.1. ábrán láthatunk példát.

Tulajdonság	Leírás
Kiterjedés	Egy rendszer kiterjedése (a lefoglalt összes terület) a komponensek összeállításától és összekapcsolásától függ.
Megbízhatóság	A rendszer megbízhatósága a komponensek megbízhatóságától függ, de nem várt kölcsönhatások új típusú hibákat eredményezhetnek, és ez kihat a rendszer megbízhatóságára.
Biztonság	A rendszer biztonsága (a támadások elleni védettség) egy olyan összetett tulajdonság, amit nehéz mérni. A támadások ott következhetnek be, ahol a rendszertervező nem várja őket, és így a beépített védelem csődöt mondhat.
Javíthatóság	Ez a tulajdonság azt tükrözi, milyen könnyen lehet behatárolni egy felfedezett problémát. Függ a probléma észlelésétől, a hibázó komponens azonosításától és a komponens lecserélésétől vagy módosításától.
Használhatóság	Ez a tulajdonság arra vonatkozik, hogy milyen könnyű használni a rendszert. Függ a technikai rendszerkomponensektől, az operátoroktól és a működési környezettől.

2.1. ábra. Példák eredendő tulajdonságokra

Az eredendő tulajdonságok két nagy típusát különböztetjük meg:

1. A *funkcionális tulajdonságok* akkor jelennek meg, amikor a rendszer minden része együttműködik valamilyen cél érdekében. Például egy biciklinek funkcionális tulajdonsága, hogy szállítóeszköz, de csak az után, hogy alkatrészeit egyszer összeszereltük.
2. A *nemfunkcionális tulajdonságok* kapcsolatban állnak a rendszer működési környezetében megfigyelhető viselkedésével. Ilyenek például a megbízhatóság, a teljesítmény, a biztonságosság és a védelem. Ezek gyakran kritikusak a számítógép-alapú rendszerek működése szempontjából, például ha ezeket a tulajdonságokat legalább valamilyen minimális szinten nem definiáljuk, a rendszer használhatatlanná válhat. Nem minden rendszerfunkció szüksé-

ges minden felhasználó számára, azaz a rendszer használható azok nélkül is, mindazonáltal az ilyen hiányos, megbízhatatlan rendszereket az összes felhasználó együttese nagy valószínűséggel nem fogadja el.

Az eredendő tulajdonságok összetettségét illusztrálandó, tekintsük a rendszer megbízhatóságát. A megbízhatóság összetett fogalom, melyet leginkább a rendszer szintjén, nem az egyedi komponensek szintjén szokás vizsgálni. A rendszer komponensei egymással szorosan összefüggnek, így az egyik komponens meghibásodása szétterjedhet a rendszerben, és hatással lehet más komponensek működésére is. A legtöbb esetben nem láthatjuk előre, hogyan terjedhetnek át a meghibásodások következményei a teljes rendszerre, így nem adhatnak megbízható becslést a rendszer komponenseinek megbízhatósági adataiból a teljes rendszer megbízhatóságára.

A rendszer átfogó megbízhatóságát meghatározó három szorosan összefüggő befolyásoló tényező:

1. *Hardvermegbízhatóság.* Mi a hardverkomponensek meghibásodásának valószínűsége és mennyi időbe telik egy ilyen komponens javítása?
2. *Szoftvermegbízhatóság.* Mennyi az esélye annak, hogy egy szoftvertermék inkorrekt kimenetet szolgáltat? A szoftvermeghibásodások a legtöbb esetben különböznek a hardvermeghibásodásoktól, amennyiben a szoftver nem megy teljesen tönkre, hibás eredmény produkálása után akár tovább is folytathatja működését.
3. *Operátormegbízhatóság.* Mennyi az esélye annak, hogy a rendszer egyik kezelője hibát vétsen?

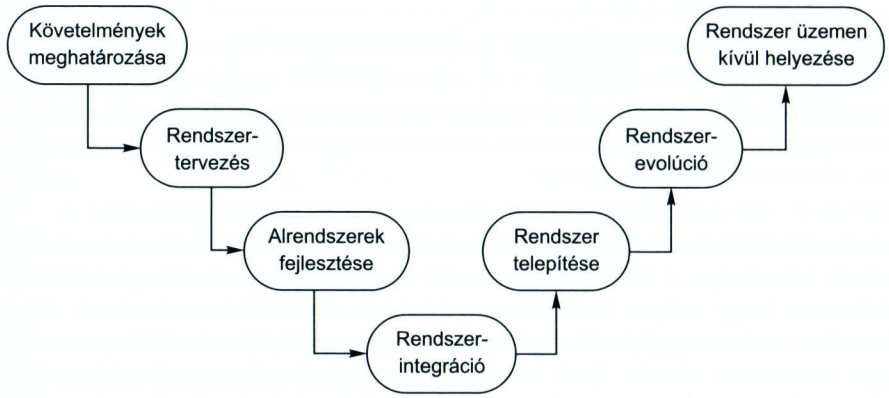
Ezek a tényezők szoros kapcsolatban állnak egymással. A hardvermeghibásodások hibás jeleket generálhatnak a rendszerben, amelyek kívül eshetnek a szoftverek által várt bemeneti tartományokon. A szoftverek ezek után előre nem várt módon működhetnek. Az operátori hibák sokkal inkább a stressz függvényei. Az ilyen operátorhibák megterhelhetik a hardvert, ami újabb hibákat eredményezhet stb. Ily módon előfordulhat az az eset, hogy egy egyszerű, könnyen javítható hiba meghibásodások sorozatát indítja el és a teljes rendszer leállásához vezethet.

A megbízhatósághoz hasonlóan, az egyéb eredendő tulajdonságok is, mint például a teljesítmény vagy használhatóság megbecslése is nehéz feladat, viszont a rendszer üzembeállítása után ezek már mérhetők. Az olyan tulajdonságok, mint a biztonságosság vagy a védelem, különféle problémákat vetnek fel, ugyanis nem egyszerűen egy olyan tulajdonsággal kell foglalkoznunk, mely a teljes rendszer viselkedésével kapcsolatban áll, hanem olyan viselkedésekkel, melyeket a rendszer *nem* mutat meg. A rendszer védett, ha nem engedi meg, hogy jogosulatlanul hozzáférhessünk adataihoz, de belátható, hogy lehetetlen meghatározni minden lehetséges elérési módot, explicit megtiltanunk azokat, és csak az alapértelmezettet engedélyezni. Amit tudunk az az, hogy egy rendszer rosszul védett, ha valakinek sikerül azt feltörnie.

2.2. RENDSZERTERVEZÉS

A rendszertervezés nem más, mint a szociotechnikai rendszerek specifikációs, tervezési, implementációs, validációs, üzembe állítási és karbantartási tevékenységeinek összessége. A rendszertervezők nem csak a szoftverekkel foglalkoznak, hanem a hardverrel és a rendszer felhasználókkal való kapcsolatával is. Sőt gondolniuk kell majd a rendszer által biztosított szolgáltatásokra, valamint a rendszer működési környezetének megszorításaira is. Mint ahogy azt már kifejtettem, a szoftvertervezőknek meg kell érteniük a rendszertervezőket, ugyanis gyakori, hogy a szoftvertervezéskor felmerülő problémák rendszertervezési döntések eredményei (Thayer, 1997; 2002).

A rendszertervezési folyamat fázisait a 2.2. ábra mutatja. Ez a folyamat nagy hatást gyakorolt a „vízesés" modellre, amelyről a 4. fejezetben szólok.



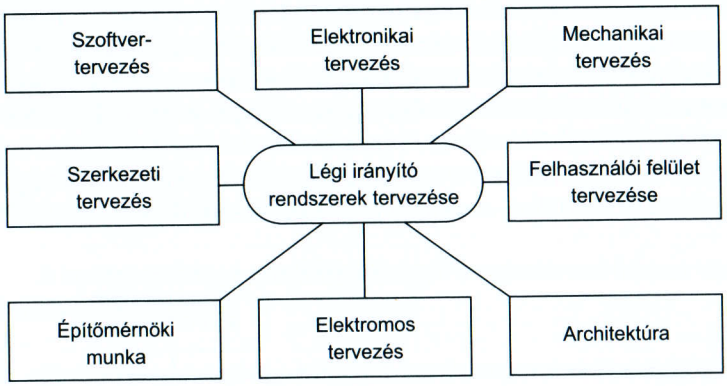
2.2. ábra. A rendszertervezési folyamat

Hatalmas különbség van a rendszertervezési és a szoftverfejlesztési folyamat között:

1. *A fejlesztés alatt nincs, vagy alig van lehetőség az átdolgozásra.* Egy rendszertervezési döntés és kivitelezés után – például egy mobil távközlési rendszerben történő torony telepítése után – annak megváltoztatása túlságosan költséges. A rendszertervek átdolgozása az ilyen problémák megoldására csak ritkán lehetséges. Íme egy ok, amiért a rendszerekben a szoftverek olyan fontossá váltak, biztosítva az esetleges új követelményeknek megfelelő fejlesztés közbeni flexibilis változtatások lehetőségét.
2. *Interdiszciplináris vonatkozások.* Számos különböző mérnöki tudományág érintett a szoftvertervezésben, nagy teret engedve a félreértéseknek, ugyanis a különböző mérnökök különböző terminológiát használnak.

A rendszertervezés különböző hátterekkel rendelkező csapatok interdiszciplináris tevékenysége. Mivel ahhoz, hogy lássuk a rendszer tervezési dönté-

seinek minden lehetséges következményét, széles körű ismeretek kellenek, ezért rendszertervező csapatokra van szükség. Tekintsünk egy légiforgalom-irányítási rendszert, mely radarokat és egyéb érzékelőket használ a repülőgépek pozíciójának meghatározására. A 2.3. ábra néhány olyan szakterületet mutat be, amelyekhez egy rendszertervező csapatnak értenie kell.



2.3. ábra. A rendszertervezés összetettsége

Számos rendszerben szinte végtelen a különféle típusú alrendszerek egybeépítésének lehetősége. A különböző mérnöki területeket képviselőknek közösen kell eldönteni, hogy milyen funkciókat nyújtsanak. Gyakran nem hozható „korrekt" döntés arról, hogyan kell egy rendszert részekre bontani. Sőt számos lehetséges alternatíva létezik. Ezek közül valamely alternatíva kiválasztása nem szükségszerűen technikai okok alapján történik. Mondjuk egy légiforgalom-irányítási rendszerben használt alternatíva új radarok építéséhez lehet jobb, mint a régiek felújítására. Amennyiben építőmérnökök is érintettek a folyamatban, más munka hiányában ezt az alternatívát fogják favorizálni, ugyanis így lehetségesnek látszik, hogy megtartsák munkájukat. Lehet, hogy később ésszerűsítik ezt a döntést technikai érvek alapján.

2.2.1. A rendszerkövetelmények meghatározása

A rendszerkövetelmények meghatározásának feladata annak definiálása, hogy mit kell a rendszernek csinálnia (funkcionálisan), és mik a szükséges rendszertulajdonságok. Mint a szoftverkövetelmény-elemzés esetében (2. rész), a folyamat az ügyféllel, illetve a végfelhasználókkal történő konzultációt is magában foglalja. A követelménymeghatározás fázisa általában a követelmények következő három típusának felismerésére koncentrál:

1. *Absztrakt funkcionális követelmények.* A rendszer által szolgáltatandó alapfunkciók absztrakt szinten definiáltak. A funkcionális követelmények specifikáció-

jának kifejtése az alrendszerek szintjén kap helyet. Például egy légiforgalom-irányítási rendszerben felismerik egy olyan repülőgép-adatbázis szükségességét, mely az ellenőrzött légtérbe belépő összes repülőgép adatainak tárolására szolgálna. Ugyanakkor az adatbázis részleteit nem specifikálják, hacsak nem befolyásolják más alrendszerek követelményeit.

2. *Rendszertulajdonságok.* Ezekről a nemfunkcionális eredendő rendszertulajdonságokról már korábban volt szó. Olyan tulajdonságokat foglalnak magukban, mint a rendelkezésre állás, a teljesítmény, a biztonságosság stb. A nemfunkcionális rendszertulajdonságok az összes alrendszer követelményeire hatással vannak.
3. *Jellemzők, amelyekkel a rendszernek nem kell rendelkezni.* Néha ugyanolyan fontos definiálni, mi az, amit a rendszernek nem kell tudnia, mint azt, hogy mit kell megtennie. Például egy légiforgalom-irányítási rendszer esetében specifikálni kell, hogy a rendszer a vezérlőt nem fogja túl sok információval ellátni.

A követelmények meghatározásának egyik legfontosabb része, hogy ki kell alakítanunk az átfogó célok halmazát, és a rendszernek javallott ezeknek eleget tennie. Nem szükségszerű ezeket a rendszer funkciói szemszögéből megadni, de azt ajánlatos definiálni, hogy a rendszer miért követel meg egy bizonyos környezetet.

A különbség érzékeltetésére nézzük egy hivatali épület tűz- és betörésjelző rendszerét. A rendszer funkcióin alapuló célok a következő állításként fogalmazhatók meg:

> *Az épületben a riasztórendszernek belső, illetve külső figyelmeztetést kell szolgáltatnia tűz vagy jogosulatlan behatolás esetén.*

Ezek a célok egyértelműen azt állítják, hogy egy olyan riasztórendszer szükséges, amely képes figyelmeztetni a nem kívánt események bekövetkezésekor. Hasonló állítás meghatározása szükséges abban az esetben, ha már egy meglévő riasztórendszer helyettesítéséről van szó. Ezzel ellentétben a célok tágabb meghatározása a következő:

> *Biztosítani kell az épületen belül folyó munka normális menetét, azt nagymértékben nem akadályoztathatják olyan események, mint tűz vagy jogosulatlan behatolás.*

A célok ilyen módon történő meghatározása egyaránt kibővíti, illetve leszűkíti bizonyos tervezési lehetőségek alkalmazhatóságát. Megengedi például, hogy a behatolásvédelem különböző kifinomult zártechnológiát alkalmazzon anélkül, hogy belső riasztás történne. Viszont kizárja az önműködő tűzoltókészülék alkalmazását, ugyanis az kihatással lehet az épületen belül található elektromos berendezésekre, ezáltal nagymértékben akadályozva a bent folyó munkát.

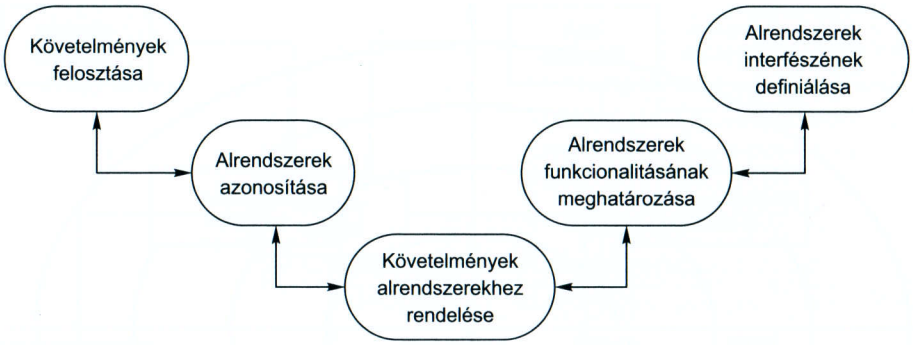
A rendszerkövetelmények meghatározásának alapvető nehézsége az, hogy azok a problémák, melyek megoldására összetett rendszereket építünk fel, gyakran úgynevezett „gonosz problémák" (Rittel és Webber, 1973). A „gonosz

probléma" olyan probléma, amely túlságosan összetett, abban sok egymással összefüggésben lévő entitás található, s így nincs pontos problémaspecifikáció. A probléma valódi természete csak akkor merül föl, amikor a megoldást már kifejlesztettük. Szélsőséges példa ilyen „gonosz problémára" például a földrengés „tervezése". Senki sem képes pontosan meghatározni, hol lesz egy földrengés epicentruma, mikor fog bekövetkezni, vagy hogy milyen hatást fog kifejteni lokális környezetére. Így azt sem tudjuk teljes egészében specifikálni, hogyan kell egy jelentős földrengést kezelni. A probléma megoldásának csak akkor láthatunk neki, miután az már bekövetkezett.

2.2.2. A rendszer tervezése

A rendszer tervezése (2.4. ábra) azzal foglalkozik, hogyan lehet biztosítani a rendszer funkcionalitását különböző komponensek segítségével. Ehhez a folyamathoz a következő tevékenységek tartoznak:

1. *A követelmények felosztása.* Az elemzett követelményeket összekapcsolódó csoportokba gyűjtjük. Általában több felosztási opció létezik, és a folyamatnak ebben a szakaszában számos alternatívát alkalmazhatunk.
2. *Az alrendszerek azonosítása.* Különböző alrendszerek azonosíthatók, amelyek egyenként vagy csoportosan megfelelnek a követelményeknek. A követelménycsoportok általában alrendszerekhez köthetők, így ez a tevékenység és a követelmények felosztása egymásba olvasztható. Az alrendszerekre osztást egyéb külső szervezetek, illetve környezeti tényezők is befolyásolhatják.
3. *A követelmények alrendszerekhez rendelése.* A követelmények hozzárendelhetők alrendszerekhez. Elvben ennek egyszerűnek kell lennie, ha a követelmények szétosztási folyamatánál figyelembe vettük az alrendszerek azonosítását is. Gyakorlatilag azonban soha nincs tiszta illeszkedés a felosztott követelmények, illetve az azonosított alrendszerek között. A külső megrendelők számára fejlesztett rendszerek megkötései gyakran azt jelentik, hogy meg kell változtatnunk a követelményeket.
4. *Az alrendszerek funkcióinak meghatározása.* Az adott alrendszerek mindegyikének adott funkciókat kell betöltenie. Ezt akár úgy is tekinthetjük, mint a rendszer tervezésének egy szakaszát, vagy ha az alrendszer szoftverrendszer, akkor mint a rendszerkövetelmények specifikációjának részét. Az alrendszerek közötti kapcsolatok felismerése és azonosítása is ebbe a szakaszba kerülhet.
5. *Az alrendszerinterfészek definiálása.* Ez azokat az interfészdefiníciókat foglalja magában, amelyeket az alrendszerek megkívánnak, illetve a későbbiekben nyújtanak. Amint egy ilyen interfészdefiníciót lezárunk, lehetségessé válik az alrendszerek párhuzamos fejlesztése.



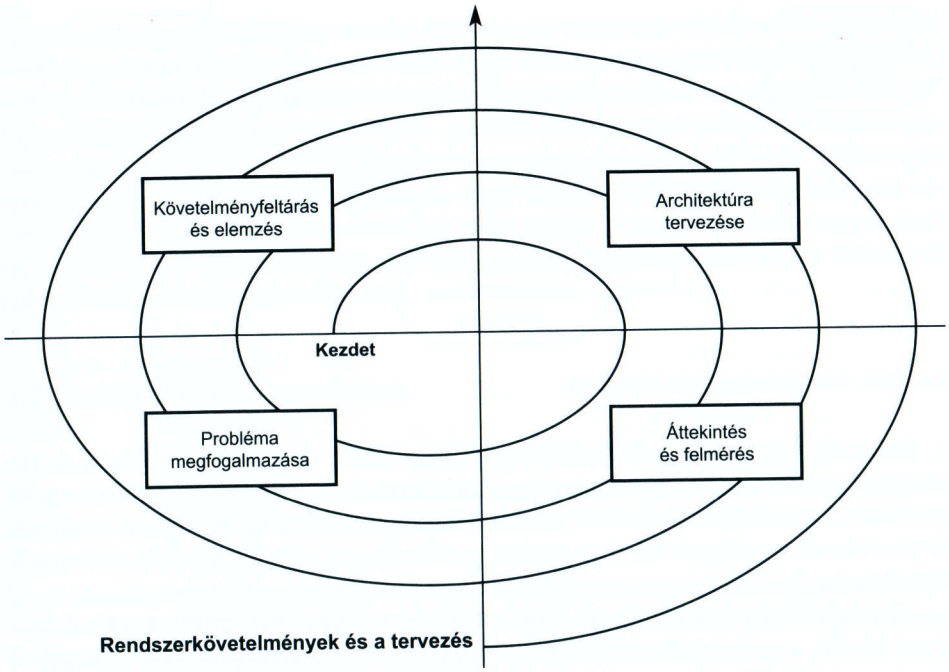
2.4. ábra. A rendszertervezés folyamata

Mint ahogy azt a mindkét oldalán nyíllal ellátott vonalak mutatják a 2.4. ábrán, nagyszerű lehetőség van a visszacsatolásra és az iterációra, amikor egyik szakaszból a másikba lépünk át a tervezési folyamatban. Amennyiben kérdések és problémák merülnek föl, úgy gyakran szükségessé válhat a korábbi szakaszok átdolgozása is.

Habár különbséget szoktunk tenni a követelményelemzés és a tervezés folyamata között, a gyakorlatban kibogozhatatlanul össze vannak fonódva. A meglévő rendszerekből fakadó megszorítások behatárolhatják a tervezési lehetőségeinket, és ezek megjelenhetnek a követelményekben is. Ahogy a tervezési folyamat folytatódik, felfedezhetünk hibákat a meglévő követelményekben, és újabb követelmények is megjelenhetnek. Következésképpen ezeket a lépéseket leginkább egy spirálként szemlélhetjük, mint ahogy azt a 2.5. ábra is mutatja.

A spirálfolyamat tehát azt mutatja, hogy a követelmények kihatással vannak a tervezési döntésekre, és viszont. A középpontból kiindulva, minden egyes körben újabb részletei kerülnek elő a követelményeknek és a tervezésnek. Bizonyos számú körök a követelményre, mások pedig a tervezésre fokuszálnak. Ha újabb ismereteket szerzünk a követelményelemzési és tervezési folyamat közben, az végül is azt eredményezi, hogy megváltozik maga a kezelendő probléma is.

Majdnem minden rendszer esetében számos lehetséges, a fejlesztésben alkalmazható tervezés létezik. Ezek különféle kombinációi megoldások széles skáláját fedik le a hardver-, a szoftver-, illetve az emberi műveletek terén. A további fejlesztésekhez választott megoldás a legmegfelelőbb olyan technikai megoldás is lehet, amely megfelel a követelményeknek. A megoldás kiválasztását nagyon sok esetben magasabb rendű szervezeti és politikai elvek befolyásolják. Ha például egy kormányzati rendszerről van szó, előfordulhat, hogy a hazai beszállítókat preferálja a külföldiekkel szemben, még akkor is, ha a hazai beszállító terméke rosszabb minőségű. Ezek főként a spirális modell áttekintés és a felmérés fázisában fejtik ki hatásukat, melyeket a követelményelemzés és a tervezés közben vagy elfogadunk, vagy elvetünk. A folyamat akkor ér véget, ha a kiértékelés azt mutatja, hogy a követelmények és a magas szintű terv sikeresen megszülettek, és elkezdhetjük a folyamat következő fázisát.



2.5. ábra. A követelményelemzés és a tervezés spirális modellje

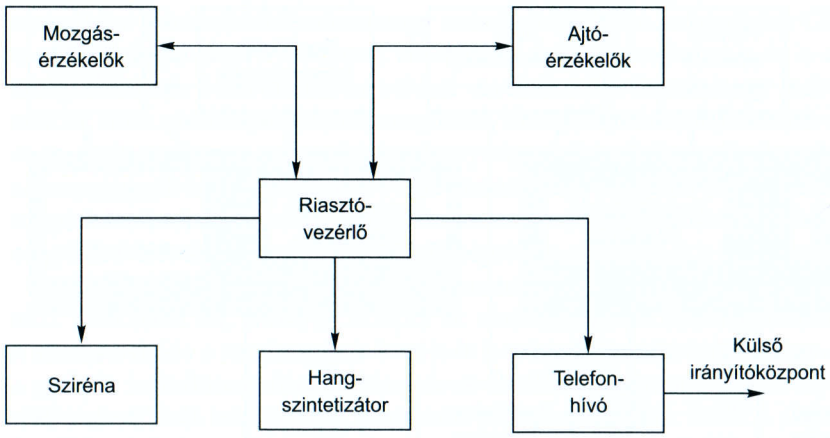
2.2.3. Rendszermodellezés

A rendszer követelményelemzési és tervezési tevékenységei közben a rendszert úgy is modellezhetjük, mint komponensek és kapcsolatok egy halmazát. Ezeket normális esetben grafikusan ábrázoljuk egy rendszerarchitektúra-modellben, mely az olvasó számára egy áttekintést nyújt a rendszer felépítéséről.

A rendszer-architektúrát általában blokkdiagrammal ábrázolják, melyből leolvashatók a főbb alrendszerek, illetve a köztük fennálló kapcsolatok. Minden egyes alrendszert egy téglalap reprezentál, két alrendszer közötti kapcsolat meglétét pedig a téglalapok közötti nyilakkal ellátott vonalak jelzik. A kapcsolatok adatfolyamok, „használt"/„használja" vagy bármilyen egyéb típusú kapcsolatok lehetnek.

A 2.6. ábra egy behatolás elleni riasztórendszer főbb komponensekre történő dekompozícióját mutatja be. A blokkdiagram kiegészíthető az alrendszerek leírásával is, amelyet a 2.7. ábra mutat be.

A leírásnak ezen a szintjén megtörtént a rendszer alrendszerekre bontása. Minden egyes alrendszer hasonló módon reprezentálható, míg a rendszert funkcionális komponensekre bontjuk. A funkcionális komponensek olyan komponensek, amelyek az alrendszerek szemszögéből nézve egyszerű feladatokat látnak el. Ezzel ellentétben az alrendszerek általában multifunkcionálisak. Természetesen



2.6. ábra. Egyszerű riasztórendszer

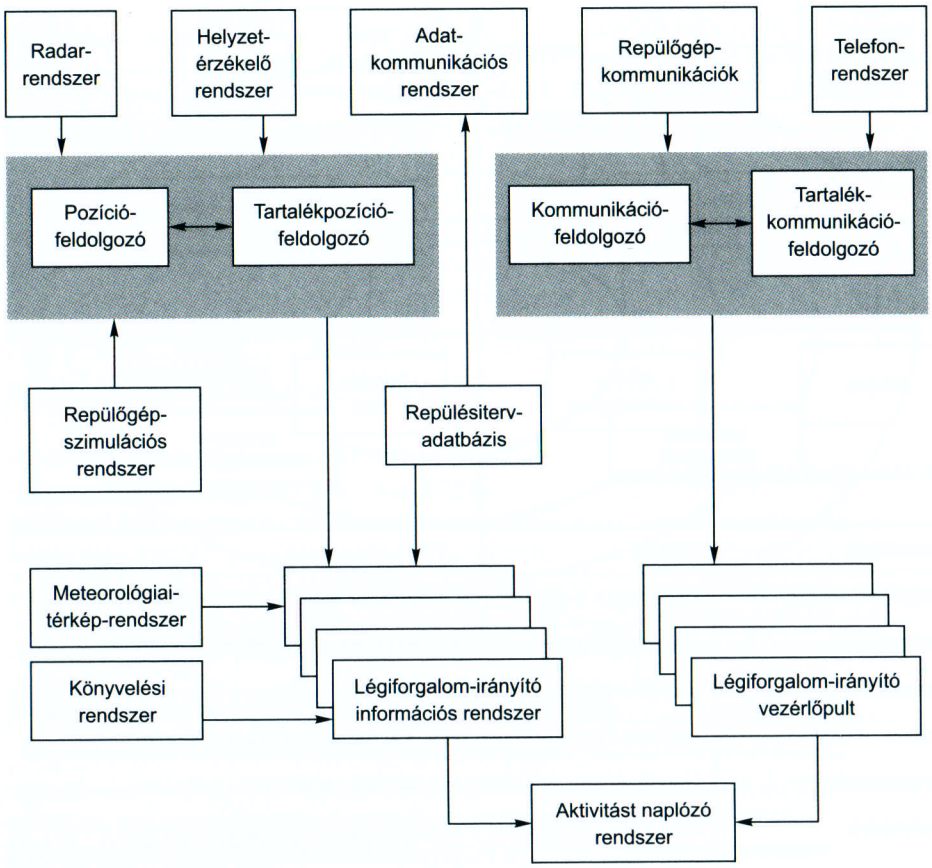
Alrendszer	Leírás
Mozgásérzékelők	Detektálják a mozgásokat a rendszer által ellenőrzött szobákban.
Ajtóérzékelők	Detektálják az épület bejárati ajtajainak nyitását.
Riasztóvezérlő	Vezérli a rendszer működését.
Sziréna	Hallható figyelmeztetést bocsát ki behatolás gyanúja esetén.
Hangszintetizátor	Hangüzenetet szintetizál, megadva a behatolás helyét.
Telefonhívó	Kimenő hívást generál a biztonsági őrség, rendőrség stb. felé.

2.7. ábra. A riasztórendszer alrendszereinek funkciói

ha egy másik perspektívából szemléljük őket (például a komponenseket gyártó cégek szemszögéből), ekkor ezek saját létjogosultsággal rendelkező rendszerek.

Történeti okokból a rendszer-architektúra modelljét a párhuzamosan fejleszthető hardver- és szoftverkomponensek azonosítására használták. Azonban ez a hardver/szoftver megkülönböztetés irrelevánssá vált. Mostanság majdnem minden komponens tartalmaz beágyazott számítási képességet. Például egy hálózati csatolóelem nemcsak kábelekből és jelismétlőkből áll, hanem hálózati kapuként, *gateway*ként is funkcionál. A jelismétlők és kapuk pedig a processzorokat és a processzorokat vezérlő szoftvereket ugyanúgy tartalmazzák, mint a speciális elektronikai komponenseket.

Architekturális szinten még helyénvalóbb, hogy az alrendszereket a funkcióikkal összhangban osztályozzuk, mielőtt döntéseket hozunk a hardver-/szoftver-optimalizáció terén. Az olyan döntéseket, hogy egy funkciót egy hardver- vagy szoftverkomponens lásson-e el, nem technikai tényezők határozzák meg, mint például COTS-komponensek elérhetősége vagy a komponens fejlesztésére rendelkezésre álló idő.



2.8. ábra. Egy légi irányítási rendszer architektúramodellje

A blokkdiagramok bármilyen méretű rendszerekre használhatók. A 2.8. ábra egy nagyobb, légiforgalom-irányítási rendszer architektúráját mutatja be. Több különböző nagyobb alrendszert fedezhetünk fel, amelyek önmagukban is nagy rendszerek. Ezek között a nagy alrendszerek közötti információáramlást a nyilakkal ellátott vonalak mutatják az ábrán.

2.2.4. Alrendszer fejlesztése

Az alrendszerek fejlesztése a rendszer tervezési fázisában meghatározott alrendszerek implementációja. Ez magában foglalhat egy másik rendszertervezési folyamatot a különálló alrendszerek számára. Ahol az alrendszer szoftverrendszer, ott egy szoftverfolyamatot lehet beágyazni, mely követelményelemzést, tervezést, implementációt stb. foglalhat magában.

Nyilvánvaló, hogy a fejlesztési folyamatban minden alrendszert az elejétől kezdünk el fejleszteni. Legalábbis rendes körülmények között, ugyanis néhány

alrendszer a kereskedelemben nagy számban kapható kulcsrakész COTS- (commercial off-the-shelf) rendszer, amelyeket megvásárolhatunk és a rendszerbe integrálhatunk. Nos, általában sokkal olcsóbb megvásárolni egy már létező terméket, mint speciális céloknak megfelelő komponenseket fejleszteni. A folyamat ezen szakaszában a tervezési tevékenységnek ismételten alkalmazkodnia kell a megvásárolt komponenshez. Nem biztos, hogy a COTS-rendszerek pontosan megfelelnek a követelményeknek, de ha az ilyen rendszerek elérhetők, általában megérik a tervek átgondolásának költségeit.

Az alrendszereket általában párhuzamosan fejlesztjük. Ha olyan problémák merülnek fel, amelyek átlépik az alrendszereink határait, módosítani kell a rendszert. Ha a rendszerünk kiterjedt hardvertervezést is magában foglal, úgy a gyártás beindítása utáni módosítások általában nagyon drágák. Gyakran „kerülőutakat" kell találnunk a probléma kompenzálására. Ezek a „kerülőutak" a legtöbb esetben szoftverváltoztatással járnak, ugyanis a szoftver a természetéből adódóan flexibilis. Ezek a szoftverkövetelmények megváltoztatásához vezetnek, így – mint ahogy azt már kifejtettem az 1. fejezetben – fontos, hogy a szoftvert úgy tervezzük, hogy felkészüljön az esetleges változtatásokra, így az új követelmények implementálása nem növeli meg túlságosan a költségeket.

2.2.5. Rendszer-integráció

A rendszer-integráció nem más, mint az egymástól függetlenül fejlesztett alrendszerek összeillesztése, hogy teljes rendszert alkossanak. Az integráció a „nagybumm" megközelítési elv szerint is kivitelezhető, azaz minden alrendszert egy adott időben integrálunk. Mindazonáltal mind technikai, mind vezetési okokból az inkrementális megközelítési elv sokkal hasznosabb az alrendszerek adoptálásához, ahol azokat egyesével integráljuk a rendszerbe.

Íme két dolog, ami alátámasztja az inkrementális folyamat előnyeit:

1. A legtöbb esetben lehetetlen úgy ütemezni a különböző alrendszerek fejlesztését, hogy a kifejlesztett rendszerek ugyanarra az időpillanatra rendelkezésre álljanak.
2. Az inkrementális integráció csökkenti a hibák lokalizációjának költségeit. Ha egy időben több alrendszert integrálunk a rendszerbe, akkor a tesztelés folyamán fellépő esetleges hiba ezek közül az alrendszerek közül bármelyikben előfordulhat. Amennyiben egyszerre csak egy alrendszert integrálunk egy működő rendszerbe, úgy az esetlegesen fellépő hiba valószínűleg az újonnan integrált alrendszerben keresendő, vagy az új és a már meglévő alrendszerek egymásra gyakorolt hatásában.

Miután egy komponens integrálásra került, a rendszer tesztelésére kerül a sor. Ennek a tesztelésnek a célja a komponensek közti interfészek tesztelése és az egész rendszer működésének tesztelése is.

Azok az alrendszer-meghibásodások, amelyek általában a más alrendszerekre tett érvénytelen feltételezések következményei, gyakran mutatkoznak meg a rendszer-integráció folyamatában. Ezek a különböző alrendszerekért felelős vállalkozók közötti vitákhoz vezethetnek. Mihelyt valamilyen problémát fedezünk fel az alrendszerek kölcsönhatásában, úgy a különböző vállalkozóknak meg kell vitatniuk, ki felelős a problémáért. A tárgyalások akár heteket, sőt hónapokat is igénybe vehetnek arról, hogyan kell a problémát megoldani.

Ahogy egyre több rendszer integrál magába COTS hardver- és szoftverkomponenseket is, a rendszer-integráció egyre fontosabbá válik. Számos esetben nincsenek szeparált alrendszerfejlesztések, és így az integráció valójában a rendszer implementációs fázisában kerül végrehajtásra.

2.2.6. A rendszer evolúciója

A nagy és összetett rendszerek nagyon hosszú élettartamúak. Élettartamuk során változnak: egyrészt korrigálni kell az eredeti rendszer követelményeinek hibáit, másrészt meg kell felelni a felmerülő új követelményeknek. A rendszerek számítógépei ez alatt az idő alatt valószínűleg kicserélődnek újabb, gyorsabb gépekre. A rendszert használó szervezetek újraszervezhetik magukat, így esetlegesen más módon használhatják a rendszert. A rendszer külső környezete megváltozhat, és ez változtatásokat hozhat létre magában a rendszerben is.

A rendszer evolúciója, hasonlóan a szoftverevolúcióhoz (21. fejezet), természetéből adódóan költséges. Ennek okai:

1. A tervezett változtatásokat mind üzleti, mind technikai szempontból gondosan elemezni kell. Emberek sorának kell jóváhagynia azokat, mielőtt hatályba helyezzük őket.
2. Mivel az alrendszerek sohasem lehetnek teljesen függetlenek, így egy változtatás egy adott alrendszerben negatív módon érintheti más alrendszerek teljesítményét vagy viselkedését. Következésképpen ezeken az érintett rendszereken is változtatásokat kell végrehajtani.
3. Az eredeti rendszerben található tervezési döntések indokait általában nem jegyzik fel, pedig azok felelőssége, hogy a rendszerevolúció szemszögéből megindokolják az adott konkrét tervezési döntéseket.
4. A rendszer öregedésével párhuzamosan jellemzően azok struktúrája is romlik a változtatások következtében, így a további változtatások költségei megnőnek.

Ahogy nő a társadalom függősége a különféle típusú rendszerektől, az evolúciónak szentelt erőfeszítések nagyobb mértékben nőnek, mint az új rendszerek fejlesztései. Ezeket a fenntartandó, meglévő rendszereket úgy hívjuk, hogy *ősrendszerek*. Az ősrendszereket a 2.4. alfejezetben tárgyalom.

2.2.7. A rendszerek üzemen kívül helyezése

A rendszerek üzemen kívül helyezése nem más, mint hasznos élettartamuk után kivonni azokat a forgalomból, felfüggeszteni az általuk nyújtott szolgáltatásokat. A hardverrendszerek esetében ez szétszerelést és az anyagok újrafelhasználását jelentheti, vagy a mérgező anyagok kivonását. A szoftvereknek nincsenek fizikai üzemen kívül helyezési problémái, de előfordulhatnak olyan esetek, hogy bizonyos szoftverek részt vesznek egy rendszer üzemen kívül helyezési folyamatában. Például a szoftverek monitorozhatják egyéb rendszerkomponensek állapotait. Mikor a rendszert üzemen kívül helyezik, a nem elhasználódott komponensek azonosíthatók és újrahasznosíthatók más rendszerekben.

Amennyiben a rendszernek olyan adatait is üzemen kívül helyezik, melyekre az adott szervezetnek még szüksége van, úgy azokat konvertálni kell, hogy más rendszerek is használhassák. Ez gyakran jelentős költségekkel járhat, ugyanis nem kizárt, hogy az adatszerkezet implicit módon magában a szoftverben volt definiálva. Elemeznünk kell a szoftvereket ahhoz, hogy feltérképezzük, hogyan vannak az adatok strukturálva, majd pedig egy programot kell készítenünk ahhoz, hogy átszervezzük az adatokat az új rendszer által igényelt szükséges formátumra.

2.3. SZERVEZETEK, EMBEREK ÉS SZÁMÍTÓGÉPRENDSZEREK

A szociotechnikai rendszerek olyan vállalati rendszerek, amelyeknek feladata az adott szervezeti, illetve üzleti célok elérése. Ez lehet akár az eladások növelése, a gyártás közben felhasznált anyagok csökkentése, az adók begyűjtése, a légtér biztonságának fenntartása stb. Mivel ezek a rendszerek egy szervezeti környezetbe vannak beágyazva, az ilyen rendszerek beszerzésére, kifejlesztésére és használatára hatással vannak az adott szervezeti előírások és eljárások, illetve az adott munkakultúra. A rendszerek felhasználói emberek, akikre hatással van a szervezet vezetése, valamint a többi, szervezeten kívüli, illetve belüli emberekkel fenntartott viszonyuk.

Ezért amikor megpróbáljuk megérteni a szociotechnikai rendszerek követelményeit, számításba kell vennünk a szervezeti környezetet is. Amennyiben ez nem történik meg, nem tudjuk kielégíteni az üzleti igényeket, és a felhasználók, valamint a vezetés elutasíthatja a rendszert.

A rendszer környezetéből származó, a rendszertervezést befolyásoló emberi és szervezeti tényezők a következőket foglalják magukban:

1. *Folyamatváltozások.* Megköveteli-e a rendszer a munkafolyamat megváltoztatását az adott környezetben? Ha igen, hirtelen továbbképzésre lesz szükség. Ha a változások jelentősek vagy az emberek munkahelyének elvesztésével járnak, fennáll a veszélye, hogy a felhasználók visszautasítják a rendszert.

2. *Feladatváltozások.* Megkövetel-e a rendszer a felhasználótól valamilyen speciális környezeti ismeretet, esetleg azt, hogy megváltoztassa munkájának addigi menetét? Ha igen, visszautasíthatják a rendszer vállalaton belüli bevezetését. Az olyan irányú elgondolások, miszerint a menedzserek meg fogják változtatni munkájukat abból a célból, hogy a számítógépes rendszereket megfelelően alkalmazzák, gyakran eleve elvetéltek. A menedzserek úgy érezhetik, a rendszerek lerombolhatják szervezeten belüli státusukat.
3. *Szervezeti változások.* Megváltoztatja-e a rendszer a politikai erőviszonyokat a szervezeten belül? Például ha a szervezet egy nagy, összetett rendszertől függ, úgy a rendszer működtetéséhez értő személyeknek jó lehetőség kínálkozik, hogy politikai hatalmat szerezzenek.

Ezek az emberi, társadalmi és szervezeti tényezők gyakran kritikus meghatározói annak, hogy egy rendszer sikeresen megfelel-e céljának vagy sem. Sajnos e tényezők rendszerekre gyakorolt hatásának megjóslása a tervezők számára hatalmas feladat, a legtöbb esetben ugyanis nem rendelkeznek megfelelő társadalmi vagy kulturális ismeretekkel, tapasztalatokkal. Segítségképpen az ilyen hatások megértéséhez rengeteg módszertan fejlődött ki, mint például a Mumford's socio-technics (Mumford, 1989) vagy a Checkland's Soft Systems Methodology (Checkland, 1981; Checkland és Scholes, 1990). Ezenkívül kiterjedt szociológiai vizsgálatokat is lefolytattak szép számmal, amelyek a számítógép-alapú rendszerek munkára gyakorolt hatásával foglalkoznak (Ackroyd *és mások*, 1992).

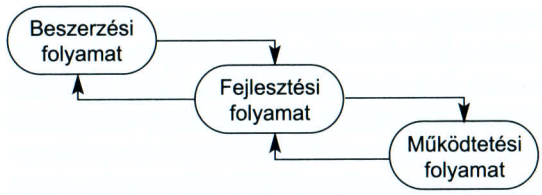
Ideális esetben minden lényeges környezeti ismeret része a rendszer-specifikációnak, így azt a rendszertervezők figyelembe vehetik a tervezés közben, de a valóságban ez sajnos lehetetlen. A rendszertervezőknek különféle feltételezésekkel kell élniük a környezetet illetően, amelyeket más rendszerekkel történő összehasonlítás, illetve közös megegyezés alapján hoznak. Ha ezeket rosszul határozzák meg, a rendszer előre nem meghatározható hibákat vétheti. Például ha egy rendszer tervezői nem ismerik fel, hogy a szervezet különböző részei eltérő célokkal rendelkezhetnek, úgy bármely általuk fejlesztett, a teljes szervezetre kiterjedő rendszer esetén előfordulhatnak elégedetlen felhasználók.

2.3.1. Szervezeti folyamatok

A 2.2. alfejezetben már szóltam a rendszertervezői folyamatmodellekről és a rendszerfejlesztés alfolyamatairól. Persze nem csak a fejlesztési folyamat jelenik meg a rendszertervezés során. Szükség van a rendszer beszerzési folyamatára, valamint a rendszer üzemeltetésének folyamatára is. Ez látszik a 2.9. ábrán is.

A beszerzési folyamat normális esetben be van ágyazva abba a szervezetbe, amelyik meg kívánja vásárolni, és használni szeretné az adott rendszert (ez a kliensszervezet). A rendszer beszerzésének folyamatában kell eldönteni, mi a legjobb módja egy rendszer beszerzésének, és ki annak a legjobb beszállítója.

A nagy összetett rendszerek kulcsrakész (COTS) alrendszerek és speciálisan elkészített komponensek keverékei. Annak oka, hogy a rendszerek mind több és

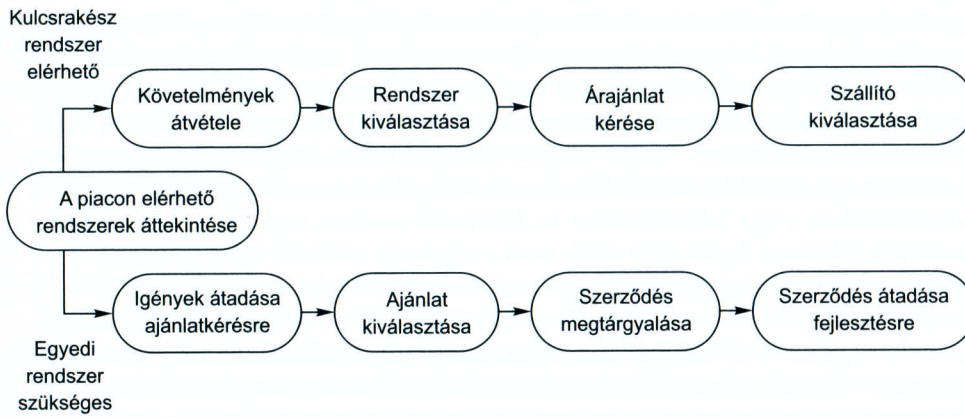


2.9. ábra. Beszerzés, fejlesztés és üzemeltetés

több szoftvert tartalmaznak, hogy ezáltal lehetővé válik a meglévő hardverkomponensek jobb kihasználása, ha a szoftvert úgy tekintjük, mint egy „ragasztót", amely ezeket a különféle hardvereket összefogja és hatékony munkára bírja. Az ilyen „ragasztó szoftver" fejlesztésének igénye az egyik oka annak, hogy a kulcsrakész komponensek használatával megtakarítások nem olyan mérvűek, mint azt várnánk. A COTS-rendszerek használatát bővebben a 18. fejezetben tárgyalom.

A 2.10. ábra a beszerzés folyamatát mutatja a létező rendszerek, illetve a speciálisan tervezett rendszerek esetében. A diagram az alábbi fontos pontokat tartalmazza a folyamattal kapcsolatban:

1. A kulcsrakész komponensek a legtöbbször nem illeszkednek pontosan a követelményeinkhez, hacsak azok nem az ilyen rendszerek szellemében íródtak. Egy rendszer kiválasztása nem más, mint megtalálni a kulcsrakész rendszer követelményeinkhez legjobban illeszkedő tulajdonságait. Ha a követelmények később megváltoznak, az kihat más alrendszerekre is.
2. Ha egy rendszert külön készítettünk el, a követelmények specifikációja adja a rendszer beszerzésére kötendő szerződés alapját, ami ily módon legalább annyira jogi, mint technikai dokumentum.
3. Miután kiválasztottuk, melyik vállalkozó készíti el a rendszert, egy tárgyalási periódus következik, ahol meg kell tárgyalni a követelmények esetleges megváltozásának lehetőségeit, és hogy azok milyen költséggel járnak.



2.10. ábra. A rendszer beszerzésének folyamata

Már korábban felvázoltam a rendszerfejlesztés folyamatának legfőbb fázisait. A komplex rendszereket általában különböző szervezetek (beszállítók) fejlesztik a rendszert beszerző szervezet számára. Ennek legfőbb oka az, hogy a vásárló alkalmazottai általában nem rendelkeznek megfelelő tudással, hogy a rendszert saját maguknak fejlesszék ki. Nagyon kevés egyedülálló szervezetnek van lehetősége arra, hogy egy nagy rendszer minden komponensét külön megtervezze, legyártsa, illetve tesztelje. Az ilyen beszállítók, akiket gyakran fővállalkozóknak hívunk, számos alvállalkozót szerződtethetnek a különböző alrendszerek kifejlesztésére. Az olyan nagy rendszerek esetén, mint például a légiforgalom-irányítási rendszer, a beszállítók egy csoportja konzorciumot alkothat, hogy ajánlatot tegyenek a szerződésre. A konzorcium már mindennel rendelkezik, ami az ilyen típusú rendszerek fejlesztéséhez szükséges, azaz van hardver- és szoftverbeszállító, szoftverfejlesztők, perifériabeszállítók vagy olyan speciális eszközök szállítói, mint például a radar.

Ez a vállalkozó/alvállalkozó modell minimalizálja azoknak a szervezeteknek számát, amelyeknek a rendszer beszerzésével foglalkozniuk kell. Az alvállalkozók megtervezik és felépítik a rendszer részeit a fővállalkozó által adott specifikációk alapján. Milhelyt azok elkészültek, azokat a fővállalkozó egy egységbe integrálja. Ezután átadhatják a rendszert a megrendelőnek megvásárlásra. A szerződéstől függően a vevő biztosíthatja a fővállalkozó számára azt a lehetőséget, hogy ő válassza meg az alvállalkozóit, vagy kötelezheti akár arra is, hogy egy előre megadott listából válassza ki azokat.

Az üzemeltetési folyamat a rendszer adott célra történő használatának folyamata. Például egy légi irányítási rendszer operátorai megadott folyamatokat követnek, amikor egy repülőgép belép a rendszerbe, és amikor elhagyja azt. Megadott tevékenységeket követnek akkor is, amikor megváltoztatják annak magasságát vagy sebességét. Az operátorokat meg kell tanítani az új rendszer használatára, ha azt szeretnénk, hogy azzal is hatékonyan tudjanak dolgozni. Itt is felmerülhetnek eddig észre nem vett problémák, amelyek a rendszer specifikációjából eredhetnek. Hiába felel meg ugyanis egy rendszer a specifikációjának, ha a specifikáció maga nem felel meg az üzemeltetési igényeknek.

Az, hogy az emberek is részét képezik a rendszernek, azzal az előnnyel jár, hogy az emberek képesek egy nem várt eseményre megfelelően reagálni, még abban az esetben is, ha nem lettek az ilyen szituációkra direkt felkészítve. Így amikor a dolgok elromlanak, az üzemeltetők gyakran meg tudják oldani a bekövetkezett szituációkat. Persze ez gyakran azzal járhat, hogy sérül a megszokott tevékenység. Az üzemeltetők felhasználhatják helyi ismereteiket a folyamatok adaptálásához vagy feljavításához is. Normális esetben ugyanis a konkrét üzemeltetési folyamat úgyis eltér attól, amit a rendszer tervezői előre terveztek.

Ez azt jelenti, hogy a tervezőknek az üzemeltetési folyamatokat flexibilisre és adaptálhatóra kell tervezniük. Az üzemeltetési folyamatok nem lehetnek túl szigorúak, azaz nem követelhetnek meg konkrét műveleteket konkrét sorrendben, vagy nem írhatják elő egy szoftver adott folyamat szerinti konkrét működését. Az üzemeltetők általában módosítanak a folyamaton, mert ők tudják, hogy mi és hogyan működik a valóságban.

Vannak még egyéb problémák, amik csak a rendszer üzembe állításakor merülnek fel, vagyis amikor elkezdi működését egy már meglévő rendszer mellett. Ilyenkor fizikai problémát jelenthet az inkompatibilitás vagy az adatok átmásolása egyik rendszerből a másikba. Sokkal kényesebb a probléma, ha a különböző rendszerek különböző felhasználói interfészekkel rendelkeznek. Egy új rendszer bevezetése megnövelheti az üzemeltetői hibák számát, mert az üzemeltetők összekeverhetik a különböző interfészek parancsait.

2.4. ŐSRENDSZEREK

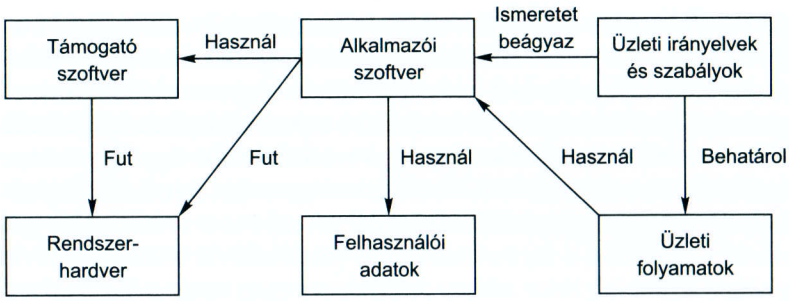
Mivel a komplex rendszerek fejlesztése időigényes és nagy erő- és anyagi ráfordításokat követel meg, ezért a nagy számítógéprendszerek életciklusa is hosszú. Például egy katonai rendszert gyakran 20 évre terveznek, és a világ légiforgalmi irányítási rendszereinek nagy részében is vannak még olyam szoftverek és üzemeltetési folyamatok, amelyeket eredetileg az 1960-as és 1970-es években fejlesztettek ki. Gyakran nagyon drága vagy nagyon kockázatos az ilyen rendszereket néhány év használat után lecserélni. Fejlesztésük tehát végighúzódik azok teljes életciklusán, hogy mindvégig megfeleljenek az új követelményeknek vagy az új üzemeltetési platformoknak stb.

Az ősrendszerek szociotechnikai számítógépes rendszerek, így magukban foglalják a szoftvert, a hardvert, az adatokat és az üzleti folyamatokat. A rendszer egyik részének megváltoztatása szükségszerűen további változtatásokat von maga után a rendszer többi komponensében. A rendszerrel kapcsolatos döntéseket nem mindig objektív tervezési kritériumok alapján hozzák, hanem azokat szélesebb szervezeti stratégiák és politikák befolyásolják.

Az ősrendszerek nagyon gyakran üzleti szempontból kritikus rendszerek. Ezért kell azokat karbantartani, mert a lecserélésük gyakran kockázatos. Például a legtöbb bank első saját rendszere általában egy ügyfél-nyilvántartási rendszer. A szervezeti irányelvek és eljárások megbíznak ebben a rendszerben. Amennyiben a bank lecseréli az ügyfélnyilvántartó rendszert (amely esetleg még drága nagyszámítógépeken fut), hatalmas kockázatot vállal, mert a lecserélés közben esetleg valami hiba történhet. Sőt a meglévő folyamatokat is meg kellene változtatni, ami felboríthatja a dolgozók addigi munkáját, és ez további problémákat is okozhat a bank életében.

A 2.11. ábra egy ősrendszer logikai összetevőit és a köztük lévő kapcsolatokat szemlélteti.

1. *Rendszerhardver.* Sok esetben az ősrendszereket nagyszámítógépekre írták, melyek ma már nem állnak rendelkezésünkre, fenntartásuk költséges, és amelyek nem felelnek meg a szervezet IT-vásárlási politikájának.
2. *Támogató szoftver.* Az ősrendszer több különböző támogató szoftverre támaszkodhat, a hardvergyártó által szolgáltatott operációs rendszertől és segéd-



2.11. ábra. Ősrendszerkomponensek

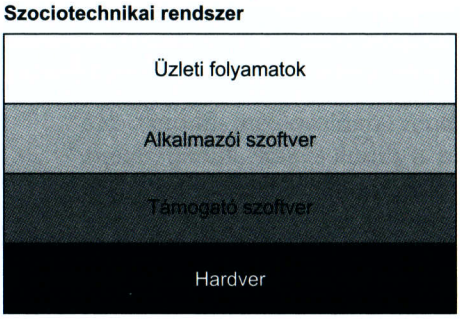
programoktól a rendszer fejlesztéséhez használt fordítókig. Ezek megint csak lehetnek elavultak, és az eredeti szállítóik többé már nem állnak mögöttük.

3. *Alkalmazói szoftver.* Mint arról a későbbiekben szó lesz, az üzleti szolgáltatásokat nyújtó alkalmazói rendszer általában több különböző programból áll össze, amelyeket nem egy időben fejlesztettek. Néha az *ősrendszer* elnevezés ezt az alkalmazói szoftverrendszert jelöli, nem pedig az egész rendszert.
4. *Felhasználói adatok.* Ezek azok az adatok, amelyeket az alkalmazói rendszer dolgoz fel. Sok ősrendszerben tömérdek mennyiségű adat halmozódott fel a rendszer használata során. Ezek az adatok lehetnek inkonzisztensek, és előfordulhat, hogy többszörösen vannak tárolva különböző állományokban.
5. *Üzleti folyamatok.* Ezek azok a folyamatok, amelyeket az üzleti célok elérésére használnak. Egy biztosítótársaságnál példa az üzleti folyamatra egy biztosítási kötvény kibocsátása; egy gyártó vállalatnál üzleti folyamat egy termékrendelés elfogadása és az ehhez tartozó gyártási folyamat előkészítése.
6. *Üzleti irányelvek és szabályok.* Ezek határozzák meg az üzleti működés kereteit. Az örökölt alkalmazói rendszer használata ezekbe az irányelvekbe és szabályokba van beépítve.

Egy ősrendszer különböző komponenseit tekinthetjük rétegek sorozatának, mint azt a 2.12. ábra mutatja. Minden egyes réteg a közvetlenül alatta lévőtől függ, és azzal van kapcsolatban. Ha az interfészeket fenntartjuk, akkor lehetséges úgy véghezvinni változtatásokat egy rétegen belül, hogy azok ne legyenek kihatással a szomszédos rétegek egyikére sem.

A gyakorlatban azonban ez az egyszerű elválasztás ritkán működik, és az egyik réteg megváltoztatása megköveteli mind a fölötte, mind az alatta lévő rétegek konzekvens megváltoztatását. Az okok a következők:

1. Egy réteg megváltoztatása a rendszerben új szolgáltatásokat hozhat be, és a rendszer magasabb rétegeit meg lehet változtatni, hogy ki tudják használni az új képességek előnyeit. Például egy új adatbázis bevitele a támogató szoftverrétegbe, magában foglalhatja azt a lehetőséget, hogy az adatokat egy webböngészőn keresztül érjük el, és az üzleti folyamatokat esetleg módosítani kell, hogy kihasználhassák ezt a szolgáltatást.



2.12. ábra. Egy ősrendszer rétegzett modellje

2. A szoftver megváltoztatása azzal járhat, hogy az egész rendszer lelassul, ezért új hardverre lesz szükség, hogy növeljük a rendszer teljesítményét. Az új hardver által okozott teljesítménynövekedés viszont azt eredményezheti, hogy további olyan szoftverváltoztatások válnak lehetségessé, amelyek korábban megvalósíthatatlanok voltak.
3. Gyakran lehetetlen a hardverinterfész megőrzése, különösen ha felvetődik egy új hardverre való radikális váltás. Például ha egy vállalat a nagygépes hardverről kliens-szerver rendszerre (a 11. fejezetben tárgyalom) tér át, mivel ezeknek általában különbözik az operációs rendszerük. Emiatt az alkalmazói szoftver jelentős megváltoztatására lehet szükség.

Kulcsfogalmak

- A szociotechnikai rendszerek magukban foglalják a hardvert, a szoftvert és az embereket is egy szervezeten belül. Azért tervezték őket, hogy segítse a szervezetet tágabb céljai elérésében.
- Az eredendő rendszertulajdonságok inkább a rendszer teljes egészének jellegzetességei, mintsem annak részeié. Ezek olyan tulajdonságok, mint például a teljesítmény, a megbízhatóság, a használhatóság, a biztonságosság és a védelem. Egy rendszer sikeres működése vagy esetleges meghibásodásai nagyban függnek ezektől az eredendő tulajdonságoktól.
- A rendszertervezési folyamat szakaszai a specifikáció, a tervezés, a fejlesztés, az integráció és a tesztelés. A rendszer-integráció során illesztjük össze a különböző beszállítók által produkált alrendszereket, és egy egészet alkotunk belőlük. Ez a folyamat nagyon kritikus.
- Az emberi és szervezeti tényezők, mint például a szervezeti felépítés és a szervezeti előírások, szignifikáns hatást gyakorolnak a szociotechnikai rendszerek üzemeltetésére.
- Egy szervezeten belül komplex kapcsolat van a rendszer beszerzési, fejlesztési és üzemeltetési folyamatai között.
- Az ősrendszer egy olyan régi rendszer, amely még mindig elengedhetetlen üzleti szolgáltatásokat nyújt.

- Az ősrendszerek nem csupán alkalmazói szoftverrendszerek, hanem szociotechnikai számítógépes rendszerek. Így magukban foglalják az üzleti folyamatokat, az alkalmazói szoftvert, a támogató szoftvert és a rendszerhardvert.

További irodalom

„Software system engineering: A tutorial". A rendszertervezés jó általános áttekintése, bár Thayer leginkább a számítógép-alapú rendszerekre fokuszál, és nem a szociotechnikai kérdésekre. (Thayer, R. H., *IEEE Computer,* April 2002.)

„Legacy information systems: Issues and directions". Áttekintés az ősrendszerek általános problémáiról, különösen az ősrendszerek adatainak problémájáról. (Bisbal, J. *és mások, IEEE Software,* September/October 1999.)

System Engineering: Coping with Complexity. Az írásának idején ez volt a legjobb elérhető rendszertervezői könyv. Ez a könyv a rendszertervezési folyamatra, a követelményekre, az architektúrára, illetve a projekt menedzselésére fektette a hangsúlyt. (Stevens, R. *és mások,* 1998, Prentice-Hall.)

„Airport 95: Automated baggage system". Kiváló és jól olvasható esettanulmány arról, hogy mit lehet rosszul csinálni egy szoftvertervezői folyamatban, és hogy ez hogyan vezethet átfogó rendszerhibákhoz. (*ACM Software Engineering Notes,* 21, March 1996.)

Feladatok

2.1. Fejtse ki, miért lehetséges, hogy egy rendszer környezetében található egyéb rendszerek nem várt hatással lehetnek a rendszer működésére.

2.2. Fejtse ki, miért nevezhetjük a katasztrófa elhárítására használatos mentőszolgálati rendszerek menedzselését természetéből adódóan *gonosz problémának*.

2.3. Határozza meg, hogyan lehet egy autóban használatos szoftverrendszer segítségünkre a teljes rendszer üzemen kívül helyezésében.

2.4. Fejtse ki, miért fontos a rendszer-specifikációs folyamat korai stádiumában megadni a rendszer architektúrájának leírását.

2.5. Tekintsünk egy biztonsági rendszert, mely a 2.6. ábrán látható rendszer kiterjesztett verziója. A rendszer a behatolásokat, illetve a tűzeseteket detektálja. Ez számítógéppel ellenőrzött füstérzékelőket, mozgásérzékelőket, ajtószenzorokat és videokamerákat tartalmaz, melyek az épület különböző pontjain lehetnek elhelyezve, valamint egy operátori konzolt, ahol a rendszer állapota követhető, illetve egy külső kommunikációs egységet is tartalmaz az esetleges tűz vagy behatolás esetén a rendőrséget, illetve a megfelelő szolgálatokat értesítendő. Rajzolja le egy ilyen rendszer lehetséges tervezésének blokkdiagramját.

2.6. Fejlesszen árvízre figyelmeztető rendszert, amely idejekorán figyelmeztet olyan esetekben, ha áradás fenyeget. A rendszer számos érzékelőt tartalmaz, amelyek monitorozzák a folyószintek változásainak mértékét, összekapcsolódik meteorológiai rendszerekkel, amelyek időjárás-előrejelzéseket szolgáltatnak, kapcsolódik a megfelelő szervezetekhez (rendőrség, parti őrség stb.) mindenféle kommunikációs eszközzel, videokamerákat tartalmaz, melyek különféle helyekre vannak felszerelve és van egy vezérlőszoba, amely operátori konzollal és videomonitorokkal van felszerelve.

A vezérlők képesek elérni adatbázis-információkat, és képesek a különféle videokijelzők között átkapcsolni. Az adatbázisrendszer különféle, a szenzorok által szolgáltatott információkat tartalmaz a kockázati tényezőknek kitett területekről és a katasztrófa lehetséges feltételeiről (például dagály, délnyugati szelek stb.), illetve a parti területek árapálytáblázatairól. Az adatbázisnak további információi lehetnek az áradást szabályozó eszközök helyeiről, azok felszereltségéről, illetve részletes információkat tartalmazhat a helyi rádióadók, illetve mentőszolgálatok elérhetőségéről.

Rajzolja le a rendszer lehetséges architektúrájának blokkdiagramját. Azonosítsa a fő alrendszereket és a köztük található kapcsolatokat.

2.7. Az Európai Múzeumok Konzorciuma egy olyan virtuális multimédia múzeum kifejlesztését kérte, amely virtuális kirándulásokkal foglalkozik az ókori Görögország idejébe. A rendszer egy 3D felületet biztosít a felhasználók számára szabványos webböngészőn keresztül, mely segítségével a felhasználható elmerülhet a virtuális valóságban. Milyen politikai és szervezeti nehézségek merülhetnek föl, amikor a rendszert telepítik a múzeumokban?

2.8. Magyarázza el, miért kritikusak az ősrendszerek az üzlet működtetésében.

2.9. Magyarázza meg, miért okozhatnak nehézséget az ősrendszerek az olyan cégeknek, amelyek meg akarják változtatni üzleti folyamataikat.

2.10. Adjon meg pró és kontra érveket a rendszertervezés mint önálló terület létjogosultságára. Érveljen annak a villamosmérnöki tervezéshez vagy szoftvertervezéshez hasonló saját létjogosultsága mellett, illetve ellen.

2.11. Tegyük fel, hogy ön egy pénzügyi rendszer fejlesztésében részt vevő tervező. A telepítés alatt felfedezi, hogy a rendszer számos ember munkáját feleslegessé fogja tenni a szervezetben. A környezetében levő emberek visszatartanak bizonyos, a teljes rendszer telepítéséhez elengedhetetlen információkat. Milyen mértékben érinti ez önt mint szoftvertervezőt? Hivatásbeli felelőssége az, hogy befejezze a rendszer telepítését, ahogy azt vállalta? Abbahagyhatja-e egyszerűen a munkát addig, amíg a megrendelő szervezet rendezi a problémát?